МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук

Кафедра программирования и информационных технологий

Веб-приложение “Wish Box”

Курсовой проект

09.03.04 Программная инженерия

Допущен к защите

Обучающийся _____М.В. Агафонова, 3 курс, д/о

Обучающийся _____А. В. Аленичев, 3 курс, д/о

Обучающийся _____Д. А. Семечев, 3 курс, д/о

Руководитель _____В.С. Тарасов, ассистент

Руководитель _____А.В. Нужных, ассистент

Руководитель _____В.А. Рыжков, ассистент

Воронеж 2020

Содержание

Содержание 2
Введение 4
1. Постановка задачи 5
2. Анализ предметной области 6
2.1 Глоссарий 6
2.2 Анализ существующих решений 6
2.2.1 Gmoji 6
2.2.2 WishBox 6
2.3 UML диаграммы 8
2.3.1 Диаграмма вариантов использования 8
2.3.2 Диаграмма последовательности 9
2.3.3 Диаграмма взаимодействия 14
2.3.4 Диаграмма состояний 18
2.3.5 Диаграмма развёртывания 19
2.3.6 Диаграмма объектов 21
2.3.7 Диаграмма активности 22
2.3.8 Диаграмма классов 23
2.4 Воронки в Яндекс Метрике. 26
2.4.1 Воронка “Создание желания” 26
2.4.2 Воронка «Оценка желаний» 27
2.4.3 Воронка «Добавление комментария» 28
2.4.4 Воронка «Отметка “Подарить”» 29
2.4.5 Воронка «Отметить как “Подарено”» 30
2.5 Схема базы данных 32
3. Обоснование архитектуры проекта 34
3.1 Определение архитектуры проекта 34
3.2 Сравнение с другими паттернами 35
4. Требования к разрабатываемой системе 39
5. Планирование работ 40

5.1 Виды работ, которые необходимо выполнить в процессе разработки программного средства 40
5.2 Состав команды, распределение задач по участникам 41
6. Реализация приложения 42
6.1 Анализ средств реализации 42
6.2 Выполнение требований по безопасности 42
6.3 Разработка API 42
7. Тестирование 46
7.1 Дымовое тестирование (smoke testing) 46
7.2 Модульное тестирование (unit testing) 48
7.3 Интеграционное тестирование (Integration testing) 48
8. Интерфейс приложения 50
8.1 Главная страница авторизованного пользователя 50
8.2 Страница «Мои желания» 50
8.3 Страница «Подарить» 51
8.4 Страница «Мои подписки» 52
8.5 Страница добавления желания 52
8.6 Страница результатов поиска 53
8.7 Страница профиля авторизованного пользователя 54
8.8 Страница редактирования профиля 54
8.9 Страница изменения пароля 55
8.10 Страница другого пользователя 56
8.11 Главная страница неавторизованного пользователя 57
8.12 Форма регистрации 57
8.13 Форма авторизации 58
Заключение 59
Список используемых источников 60
Приложение 1 61
Распределение задач по участникам команды 61
Распределение задач по участникам команды 62
Распределение задач по участникам команды 63

Введение

Все любят дарить и получать подарки. В этом есть что-то магическое. Но иногда бывает так, что полученный подарок вам не нравится, да и вы уже начинаете понимать, что подарили другому человеку не самую нужную или приятную для него вещь.

В таких случаях на помощь приходят всяческие сервисы подбора подарков, где другие люди уже составили какие-то подарочные наборы. Правда, обычно это бывает дорого и не всем по карману.

Также есть различные приложения по поиску подарков. Но хорошего приложения на данный момент не существует, а существующие не имеют нужной функциональности.

В данной курсовой работе рассматривается проблема создания веб-приложения, предоставляющего пользователям возможность публиковать список желаемых ими подарков, а также возможность дарить эти самые подарки другим людям.

1. Постановка задачи

Цель курсовой работы – реализовать web приложение, позволяющий пользователям «делиться» своими пожеланиями в качестве подарков, а также позволяющий дарить желаемый предмет. Оно должно позволять пользователю:

- создавать/редактировать/удалять своё “желание”;
- отмечать подарок другого пользователя, который они хотят ему подарить, и удалять данную отметку;
- подписываться через поиск по пользователям на другие профили и отписываться от получения новостей о добавлении новых желаний определённого пользователя;
- оставлять комментарии к записи с желанием и удалять его;
- изменять данные своего профиля, включая смену пароля.

Архитектура разрабатываемого приложения должна иметь отдельно клиентскую и отдельно серверную части. Выбор данной архитектуры обусловлен требованием отделить логику приложения и интерфейс. Должны быть реализованы следующие компоненты системы:

- база данных;
- клиентская часть приложения;
- серверная часть приложения.

2. Анализ предметной области

2.1 Глоссарий

- Желание (подарок) – запись, которую пользователь публикует на странице своего профиля в качестве отображения того, что он хотел бы получить в подарок;
- подписчик – пользователь, чьи новые желания будут видны на главной странице текущего пользователя;
- БД – база данных.

2.2 Анализ существующих решений

Рассмотрим существующие решения.

2.2.1 Gmoji

Само приложение — это витрина товаров и услуг, которые можно приобрести в подарок. Выбор большой: от кофе и цветов до мойки машины и стрижки. Всего разделов три: каталог, где можно приобрести подарки, профиль пользователя и список приобретенных и полученных подарков. Посылать подарки друзьям можно как из приложения, так и через мессенджеры, если включить в настройках клавиатуру Gmoji. Отправленные ссылки превращаются в стикеры с крупной картинкой, а в других мессенджерах выглядят просто как ссылки.

Плюсы:

- приятный дизайн и интуитивно понятный интерфейс приложения;
- огромный выбор подарков;
- не требуется наличие приложения для получателя подарка.

Минусы:

- отсутствие версии для браузеров;
- не самая удобная система “дарения” подарков через подключение еще одной клавиатуры на телефоне.

2.2.2 WishBox

Приложение доступно в AppStore и Google Play. Разрабатывает его небольшая команда разработчиков. Приложение позволяет добавлять свои

подарки, чтобы люди знали, что вам дарить, а также отмечать подарки других людей, чтобы вы не забыли

Плюсы:

- возможность создавать свои списки желаний, а также событий, к ним причастных;
- возможность добавлять людей в “друзья”;
- новостная лента, чтобы всегда видеть, чего хотят друзья;
- уведомления;
- возможность добавлять желания друга в свои.

Минусы:

- периодически происходят сбои сервера;
- интуитивно непонятный интерфейс;
- функционал на разных платформах различается;
- невозможность фильтра новостной ленты;
- невозможность фильтра результата поиска (т.е. поиск только по имени пользователя).

2.3 UML диаграммы

2.3.1 Диаграмма вариантов использования

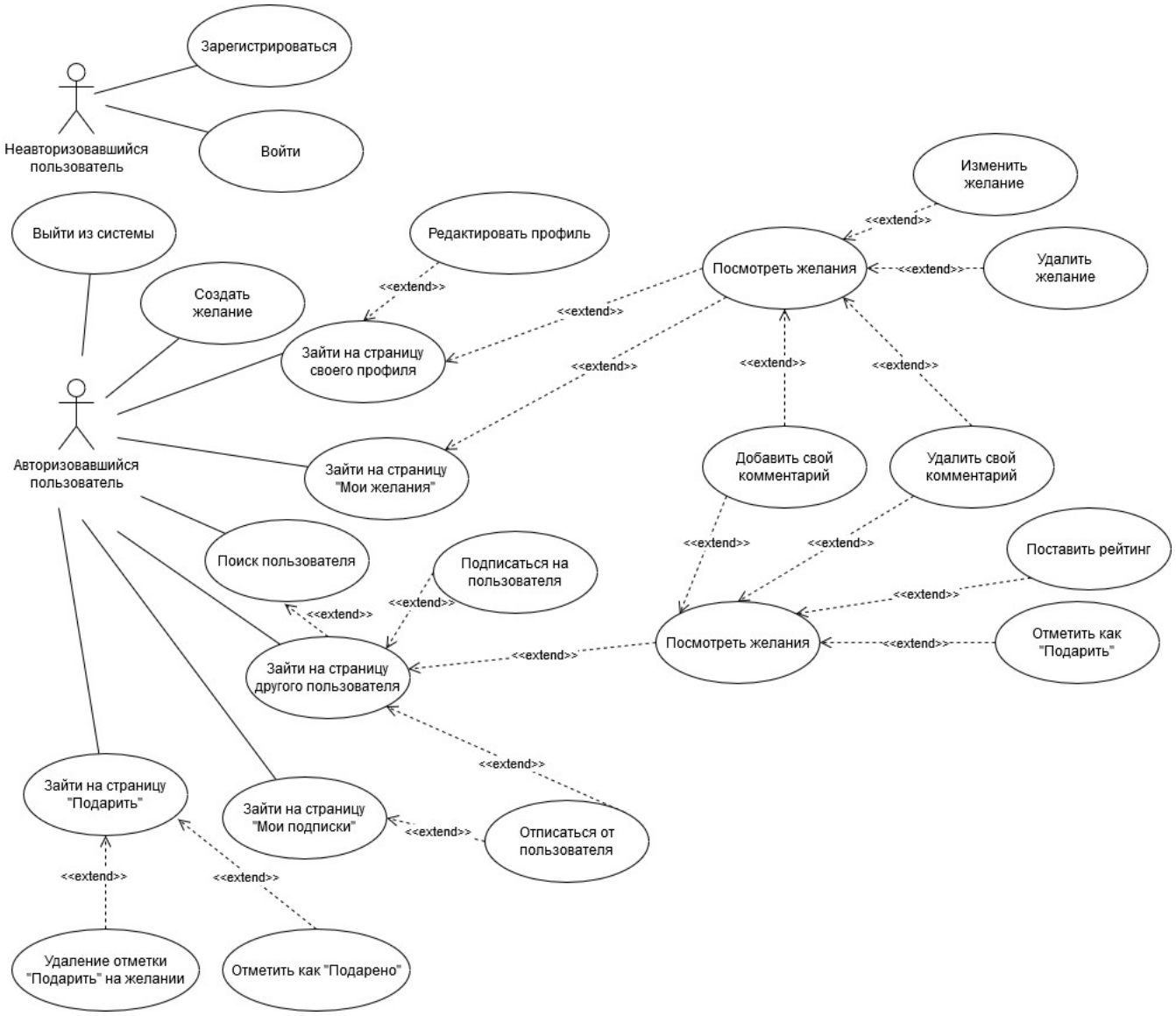

Рис. 1. Диаграмма вариантов использования

Рассмотрим диаграмму вариантов использования для приложения на рисунке 1.

В системе могут быть два вида актёра:

- неавторизовавшийся пользователь;
- авторизовавшийся пользователь.

Отношения ассоциации – действия, которые может осуществлять пользователь.

Для неавторизовавшегося пользователя отношения ассоциации:

- зарегистрироваться;
- войти.

Для авторизовавшегося пользователя отношения ассоциации:

- выйти из системы;
- создать желание;
- зайти на страницу своего профиля;
- зайти на страницу «Мои желания»;
- зайти на страницу другого пользователя;
- зайти на страницу «Мои подписки»;
- зайти на страницу «Подарить»;
- найти пользователя в системе по имени.

Для авторизовавшегося пользователя отношения расширения:

- редактировать свой профиль;
- изменить своё желание;
- удалить своё желание;
- добавить комментарий;
- удалить свой комментарий;
- поставить рейтинг желанию другого пользователя;
- поставить отметку «Подарить» на желание другого пользователя;
- удалить отметку «Подарить» у ранее отмеченного желания;
- поставить отметку «Подарено» на желание, которое есть на странице «Подарить»;
- подписаться на другого пользователя;
- отписаться от пользователя, на которого был ранее подписан текущий пользователь.

2.3.2 Диаграмма последовательности

Диаграмма последовательности для создания желания, рисунок 2

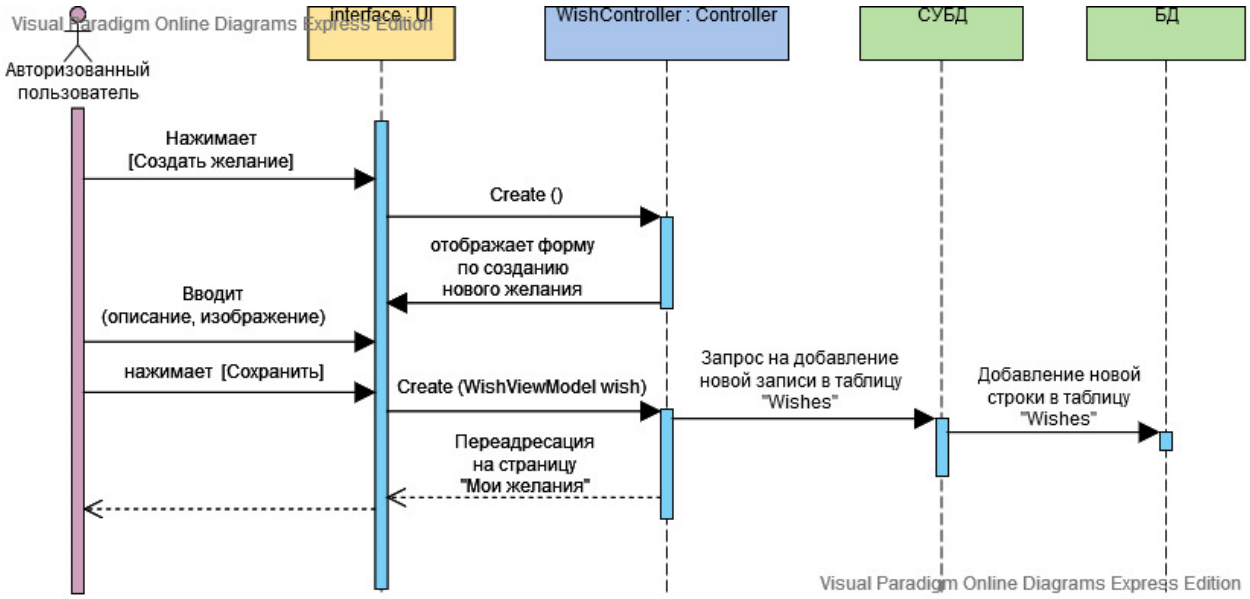

Рис. 2. Диаграмма последовательности для создания желания

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием соответствующей кнопки для создания желания) вызывается метод Create в контроллере, который возвращает форму по созданию нового желания. После введения описания желания и опциональной (необязательной) загрузки изображения для желания и нажатия кнопки «сохранить» вызывается метод Create (POST-запрос) с аргументом WishViewModel – моделью просмотра желания, которая содержит все данные, внесённые пользователем. Отправляется запрос в СУБД на добавление новой записи в таблицу «Wishes». В результате добавляется новая строка в таблицу «Wishes» в базе данных. Происходит переадресация пользователя на страницу «Мои желания».

Диаграмма последовательности для создания желания, рисунок 3

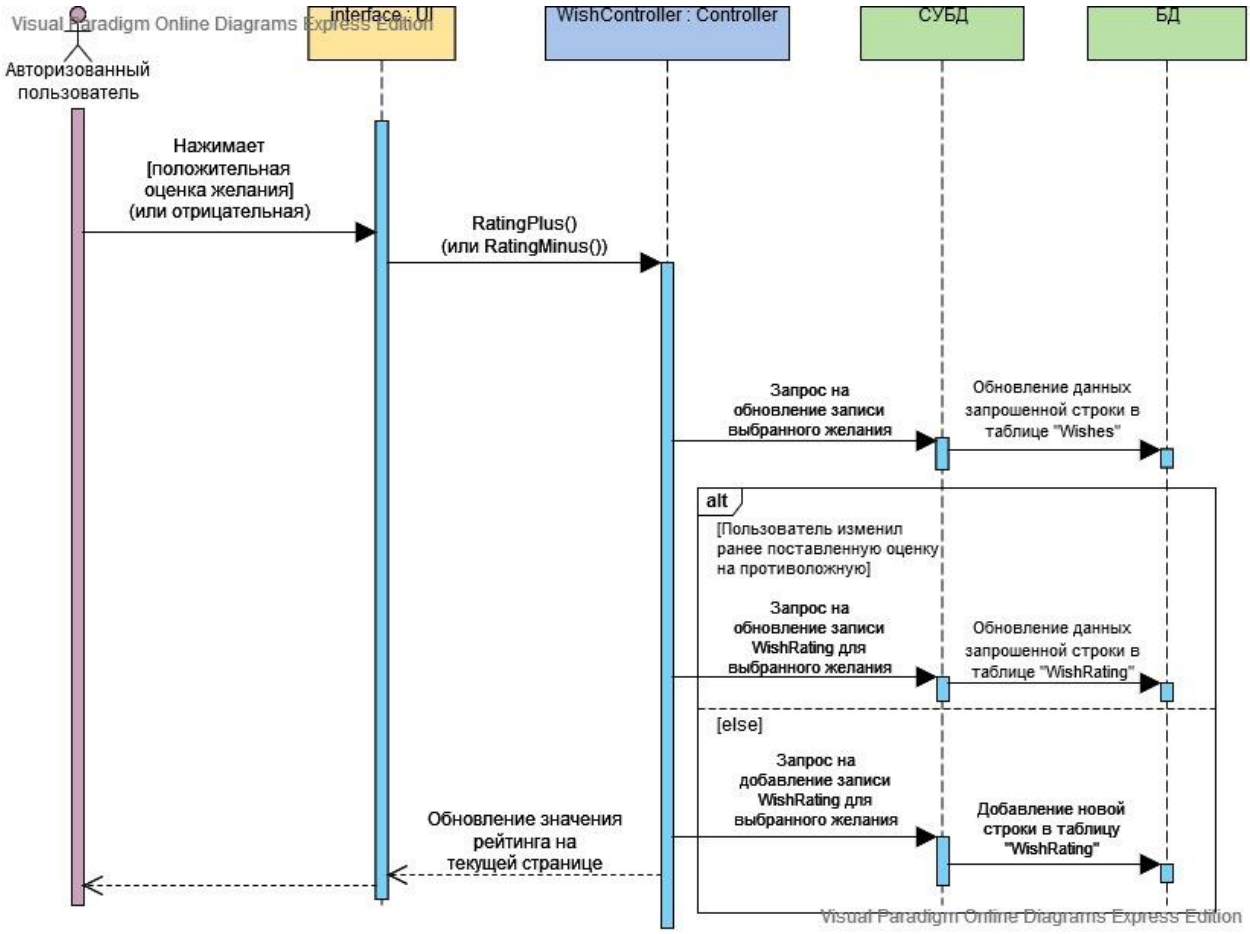

Рис. 3. Диаграмма последовательности для оценки желания

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием соответствующей кнопки для оценки желания) вызывается метод RatingPlus или RatingMinus в зависимости от выбранной оценки (положительной или отрицательной) в контроллере. Отправляется запрос в СУБД на обновление записи выбранного желания в таблице «Wishes». В результате обновляются данные строки с запрошенным WishId в таблице «Wishes» в базе данных.

Если пользователь уже ставил ранее оценку данному желанию и поставил теперь противоположную, то отправляется запрос в СУБД на обновление записи с определённым WishId в таблице «WishRating». В результате обновляются данные строки с запрошенным WishId в таблице «WishRating» в базе данных.

Если пользователь в первый раз оценил данное желание, то отправляется запрос в СУБД на добавление записи в таблице «WishRating». В результате добавляется новая строка с переданными данными в таблицу «WishRating» в базе данных.

Происходит обновление значения рейтинга на текущей странице.

Диаграмма последовательности для добавления комментария, рисунок 4

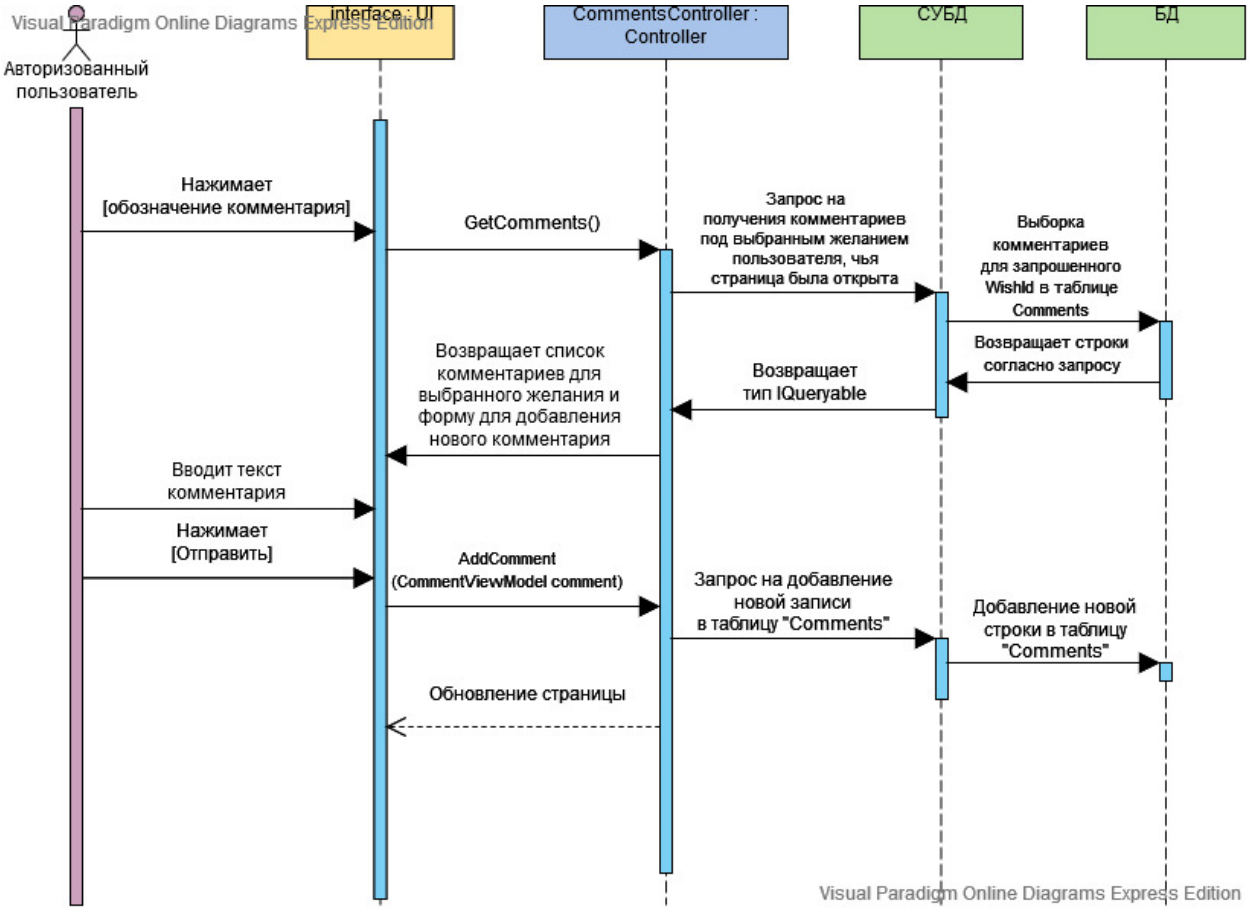

Рис. 4. Диаграмма последовательности для добавления комментария

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки, обозначающей комментарий) вызывается метод GetComments в контроллере. Отправляется запрос в СУБД на получение комментариев для выбранного желания пользователя, страница которого была открыта. В результате происходит выборка комментариев для запрошенного WishId в таблице «Comments» в базе данных. СУБД возвращается в контроллер данные типа IQueryable, т.е. множество комментариев. Полученный список отображается на странице с формой для добавления нового комментария.

После того, как пользователь ввёл содержание комментария в соответствующее поле в появившейся форме и начал кнопку «Отправить», вызывается метод AddComment в контроллере. отправляется запрос в СУБД

на добавление записи в таблице «Comments». В результате добавляется новая строка с переданными данными в таблицу «Comments» в базе данных.

Происходит обновление страницы.

Диаграмма последовательности для отметки «Подарить», рисунок 5

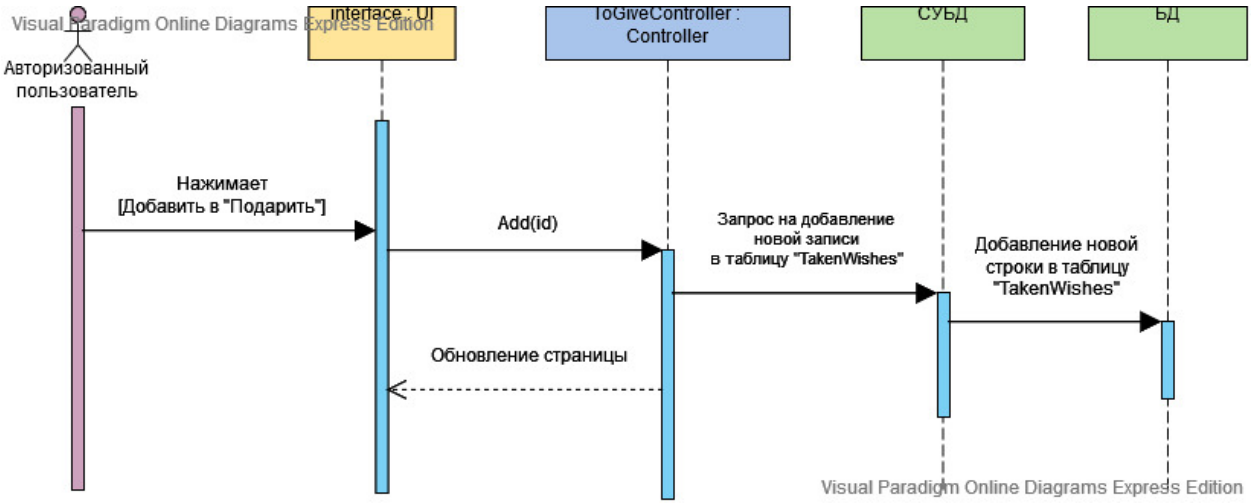

Рис. 5. Диаграмма последовательности для отметки «Подарить»

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки «добавить в “подарить”») вызывается метод Add в контроллере. Отправляется запрос в СУБД на добавление новой записи в таблицу «TakenWishes». В результате добавляется новая строка в таблицу «TakenWishes» в базе данных. Происходит обновление страницы.

Диаграмма последовательности для отметки «Подарено», рисунок 6

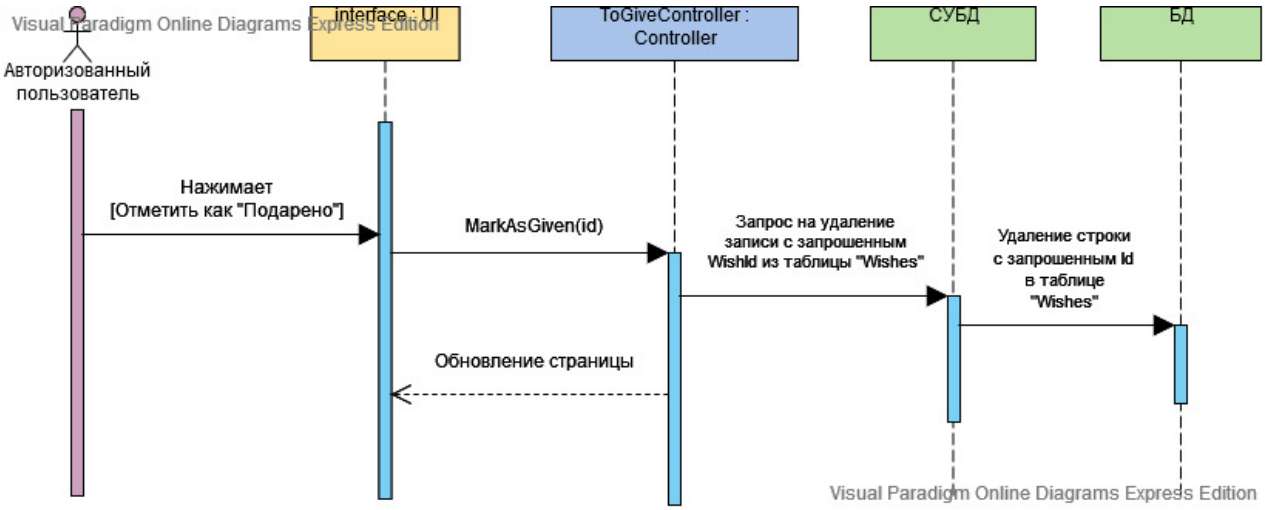

Рис. 6. Диаграмма последовательности для отметки «Подарено»

После взаимодействии пользователя, который обязательно должен быть авторизован, с интерфейсом приложения (нажатием кнопки «отметить как “подарено”») вызывается метод MarkAsGiven в контроллере. Отправляется запрос в СУБД на удаление записи с запрошенным WishId в таблицу «Wishes». В результате удаляется строка с запрошенным WishId в таблицу «Wishes» в базе данных. Происходит обновление страницы.

2.3.3 Диаграмма взаимодействия

Диаграмма взаимодействия для создания желания, рисунок 7

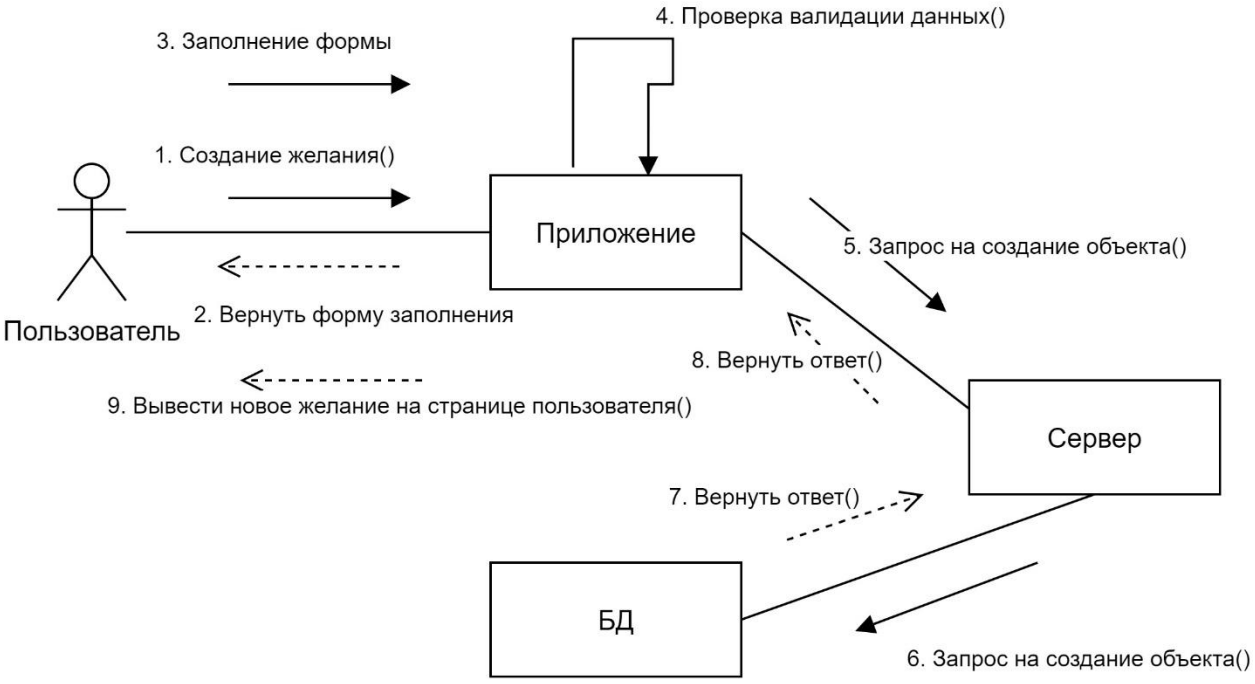

Рис. 7. Диаграмма взаимодействия для создания желания

Для добавления нового желания пользователю необходимо на главном экране нажать кнопку «Добавить желание» (обозначена как + с подарком). После этого отображается окно для ввода данных о желании. Пользователю необходимо ввести данные и нажать кнопку «Сохранить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список желаний во вкладке “Мои желания”.

Диаграмма взаимодействия для оценки желания, рисунок 8

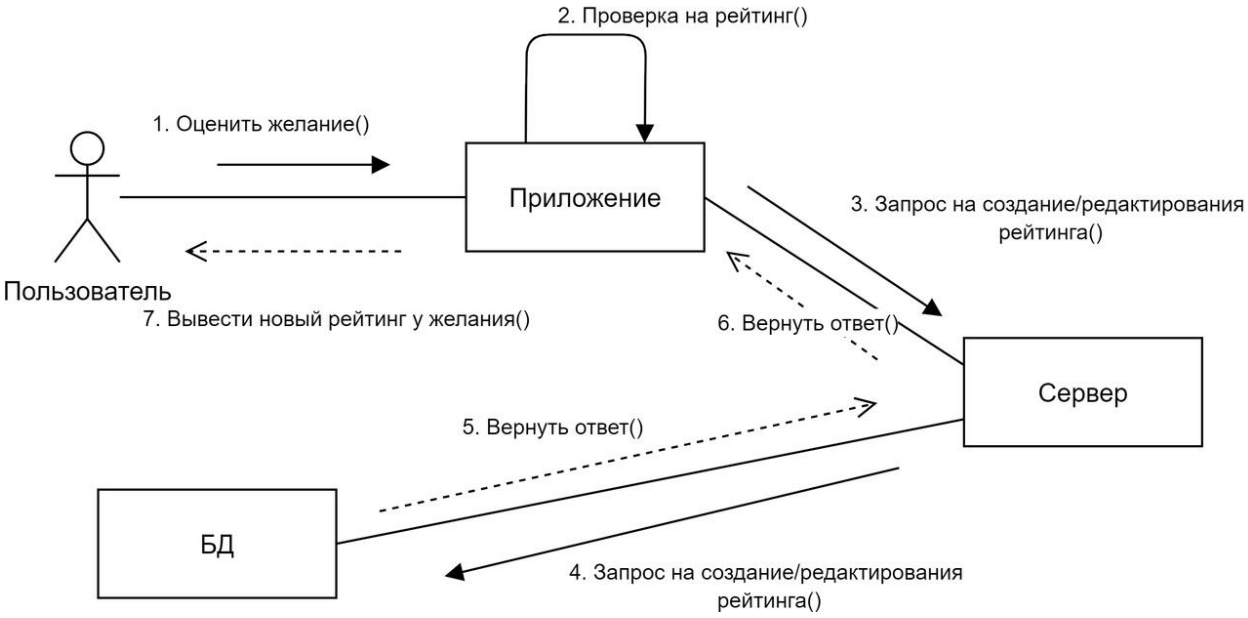

Рис. 8. Диаграмма взаимодействия для создания желания

Для оценки желания пользователю необходимо на странице пользователя, чье желание он хочет оценить нажать на кнопки “+” или “-” соответственно тому, как он хочет оценить его. После этого приложение отправляет запрос на сервер, который обращается к БД с целью вернуть предыдущее оценивание данным пользователем этого желания. Производится проверка предыдущей оценки и в зависимости от неё приложение отправляет запрос на сервер с целью создания/редактирования предыдущей оценки. Тот в свою очередь передает запрос в БД. Если все выполнено верно, то приложение получает новый рейтинг желания, который оно и выведет.

Диаграмма взаимодействия для добавления комментария, рисунок 9

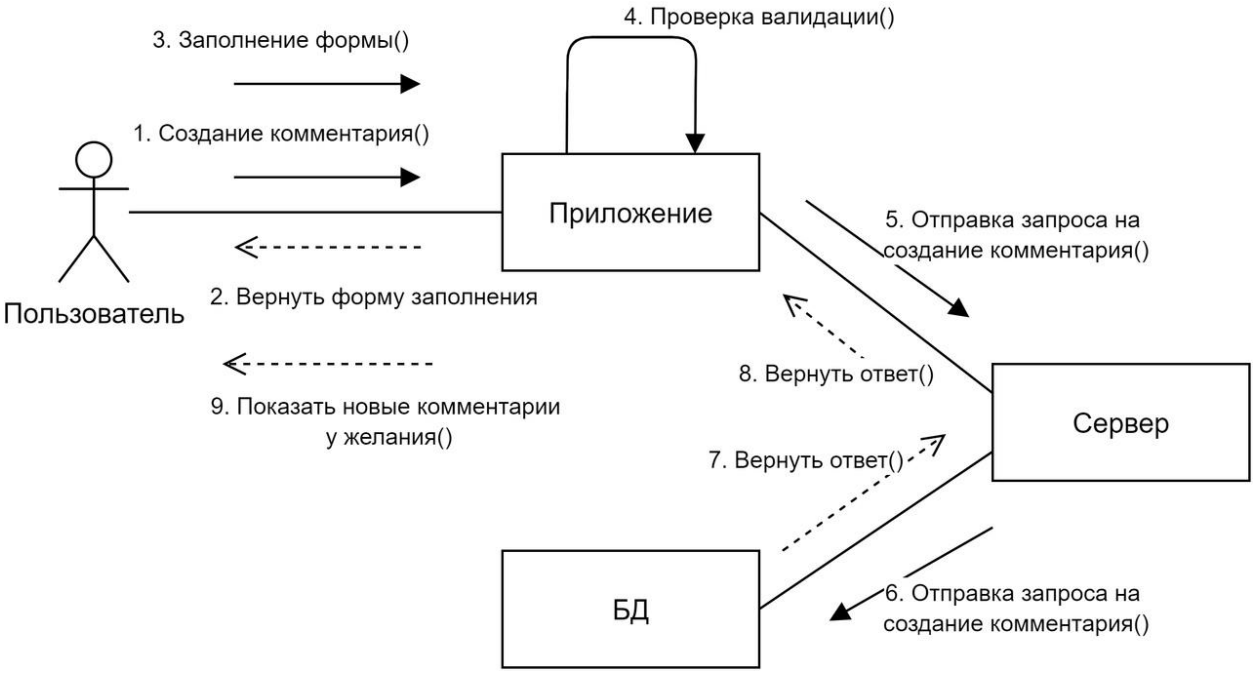

Рис. 9. Диаграмма взаимодействия для добавления комментария

Для добавления нового комментария к желанию пользователю необходимо возле желания, под которым он хочет оставить комментарий, нажать кнопку “Комментарии”. После этого отображается строка для ввода комментария. Пользователю необходимо ввести данные и нажать кнопку «Отправить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список комментариев. Если пользователю необходимо ответить на чей-то комментарий, то с начала он должен нажать кнопку “Ответить” возле того комментария, которому он хочет ответить, после чего вся вышеописанная процедура повторяется.

Диаграмма взаимодействия для отметки «Подарить», рисунок 10

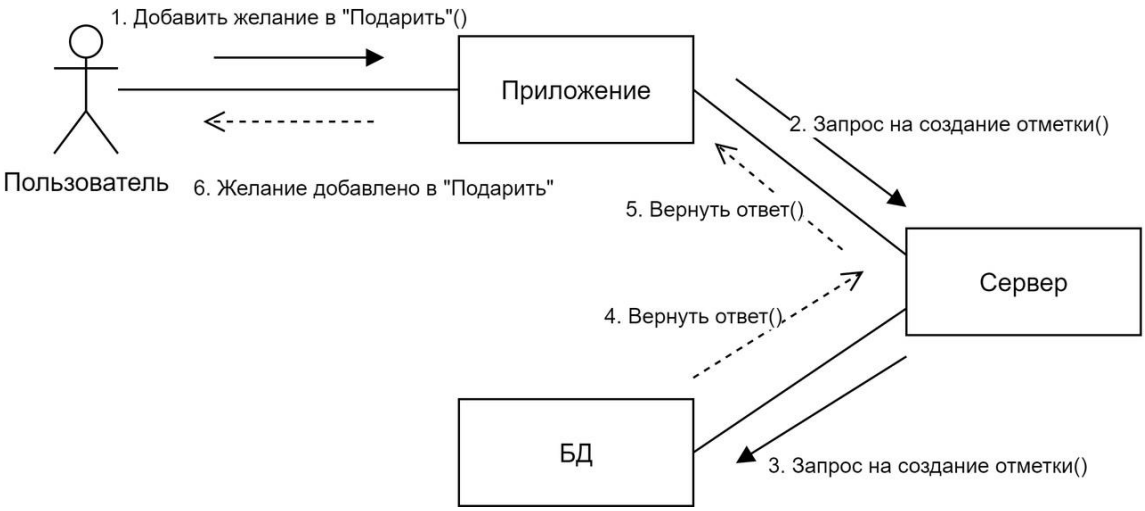

Рис. 10. Диаграмма взаимодействия для добавления комментария

Для добавления желания в “Подарить” пользователю необходимо у желания на странице того пользователя, которому он хочет подарить нажать кнопку «Добавить в Подарить». После этого приложение делает запрос на сервер, который обращается в БД с запросом на создание отметки. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список подарков во вкладке “Подарить”.

Диаграмма взаимодействия для отметки «Подарено», рисунок 11

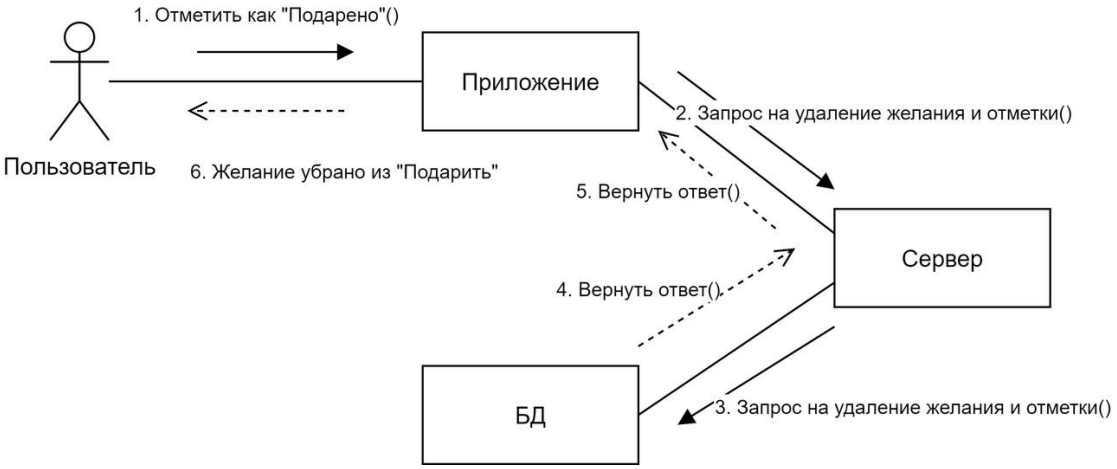

Рис. 11. Диаграмма взаимодействия для добавления комментария

Для отметки желания как “Подарено” пользователю необходимо во вкладке “Подарить” рядом с желанием, которое он подарил нажать кнопку «Отметить как “Подарено”». После этого приложение делает запрос на сервер, который обращается в БД с запросом на удаление отметки о подарке и самого желания. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список подарков во вкладке “Подарить”.

2.3.4 Диаграмма состояний

На рисунке 12 изображена диаграмма состояний.

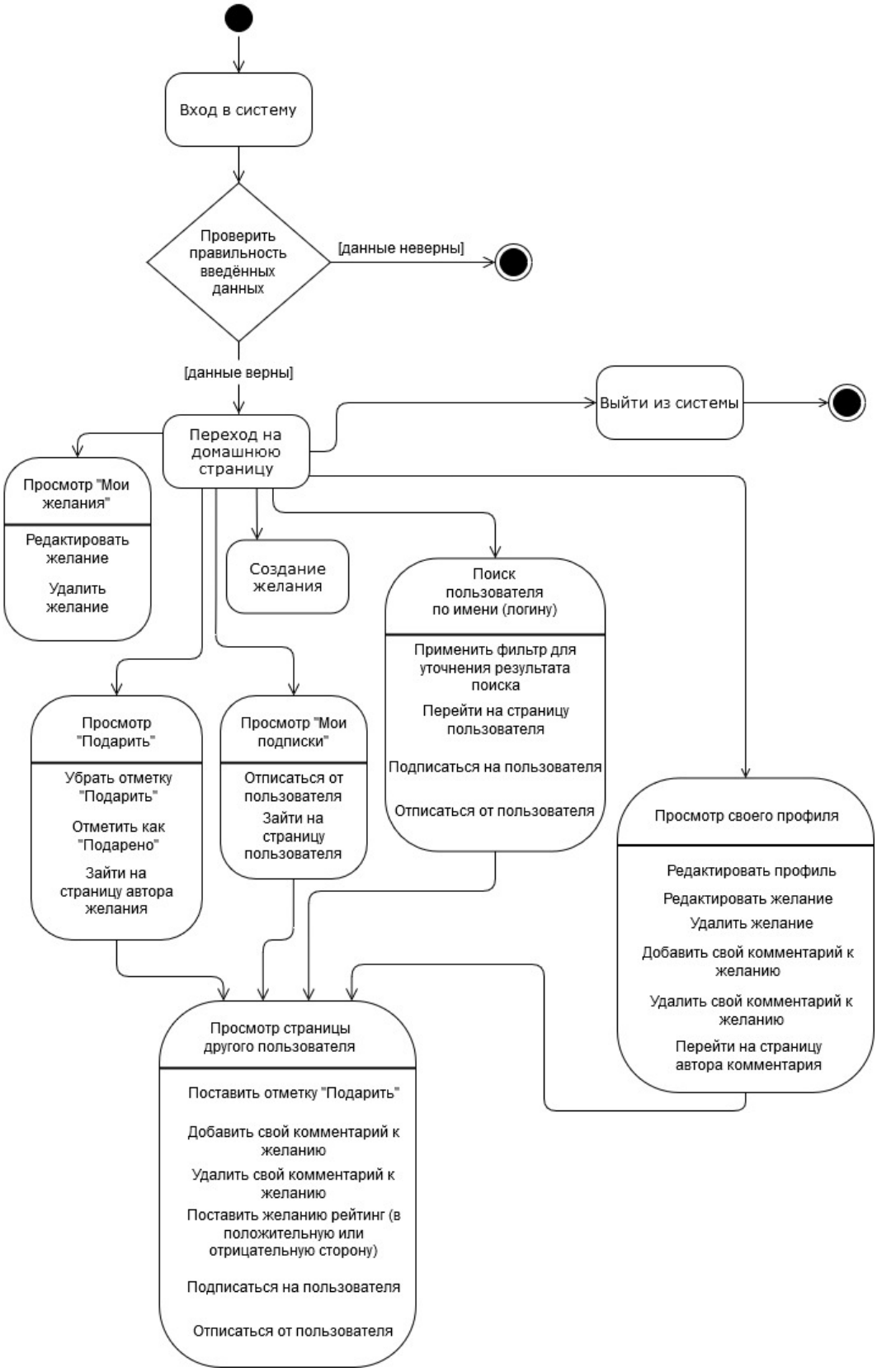

Рис. 12. Диаграмма состояний

Диаграмма состояний отражает возможные состояния системы. При входе в систему осуществляется проверка корректности данных, которые ввёл пользователь для входа. В зависимости от результата проверки возможны 2 цепочки состояний, связанных с открытием доступа к функциональности приложения для пользователя.

Если результат проверки показал, что данные верны, система переадресовывает пользователя на домашнюю страницу, где для него доступна необходимая функциональность сайта.

Если пользователь выбрал функцию показа своего профиля, система обрабатывает запрос, отображает пользователю запрошенную страницу, где текущему пользователю предоставляется соответствующий данному состоянию перечень функций.

Аналогично происходит и с другими цепочками состояний.

Из любого состояния системы, кроме входа в систему, пользователь может выйти из системы.

2.3.5 Диаграмма развёртывания

На рисунке 13 представлена диаграмма развёртывания, чтобы определить, какие аппаратные компоненты («узлы») существуют, какие программные компоненты работают на каждом узле и какие различные части этого комплекса соединяются друг с другом.

Для развёртывания приложения требуется сервер с высокоскоростным доступом в интернет. Пользователю для взаимодействия с приложением требуется персональный компьютер с доступом в интернет, а также устройствами ввода-вывода информации, такие как клавиатура и мышь или сенсорная панель и монитор (экран).

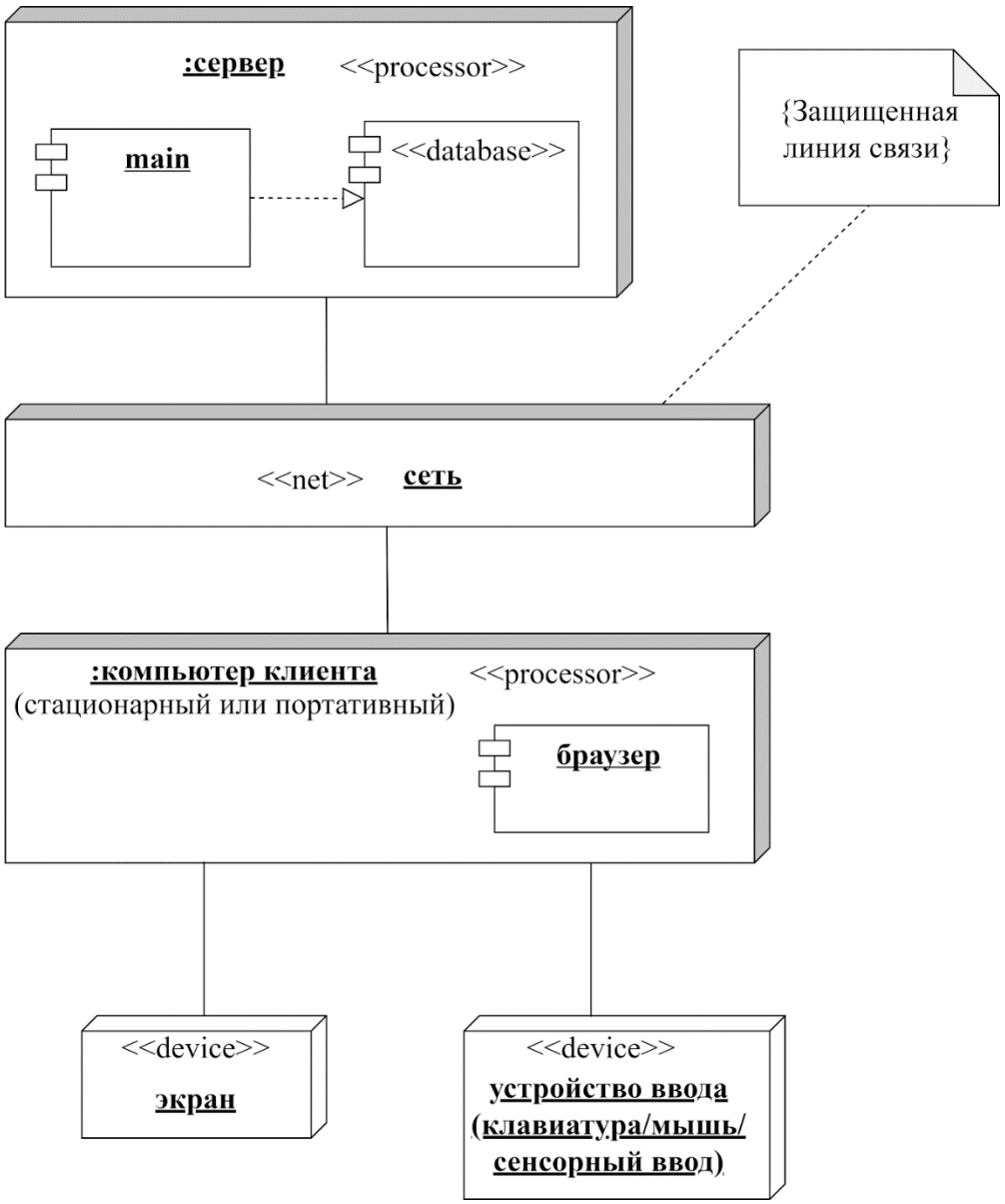

Рис. 13. Диаграмма развёртывания

2.3.6 Диаграмма объектов

На рисунке 14 представлена диаграмма объектов

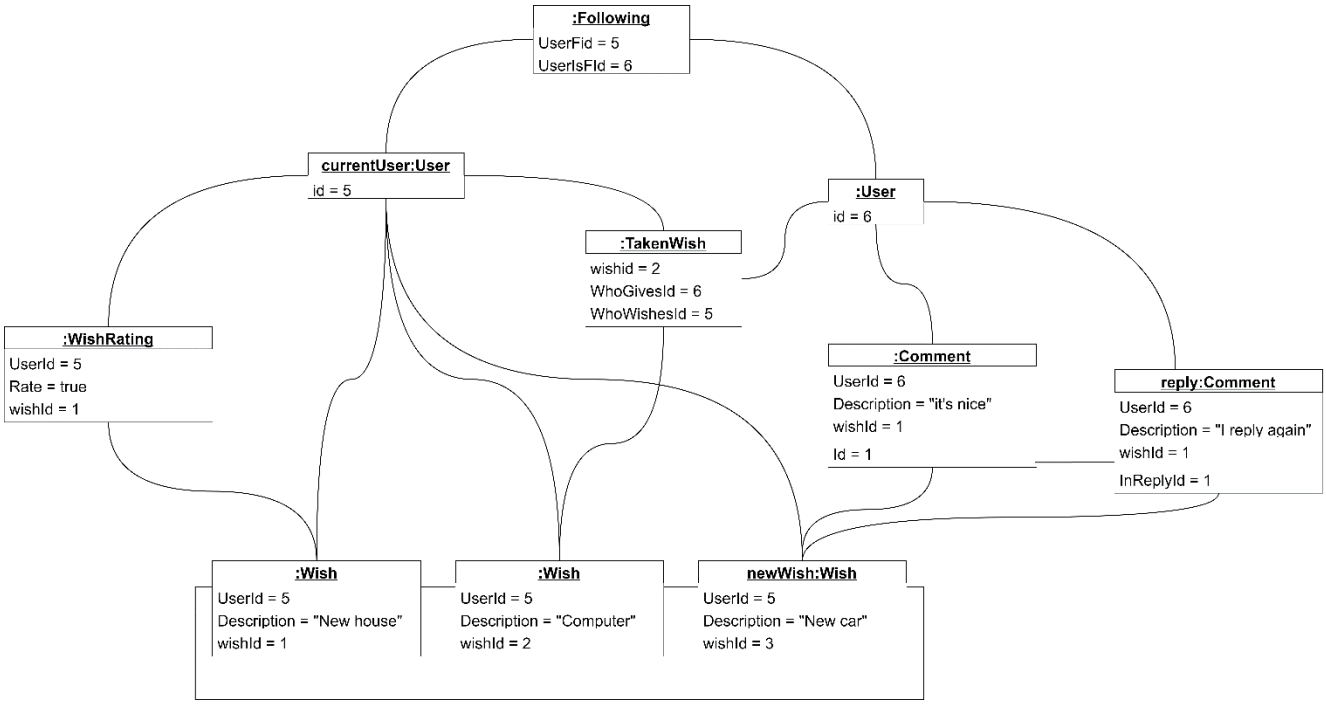

Рис. 14. Диаграмма объектов

Пусть в некоторый момент времени существуют следующие объекты:

- экземпляр класса User с именем currentUser выступающий в роли текущего авторизованного пользователя;
- 3 экземпляра “желаний” currentUser;
- анонимный экземпляр класса User;
- экземпляр класса WishRating, хранящий информацию о том, что текущий пользователь нажал на кнопку “+” рейтинга своего желания;
- экземпляр класса Following, хранящий информацию о том, что пользователь с Id = 6 подписался на новости currentUser;
- экземпляр класса TakenWish, хранящий информацию о том, что пользователь с Id = 6 решил подарить одно из желаний currentUser и нажал соответствующую кнопку;
- анонимный экземпляр класса Comment, хранящий информацию о комментарии, оставленным пользователем с Id = 6 под желанием пользователя currentUser;

2.3.7 Диаграмма активности

На рисунке 15 изображена диаграмма активности

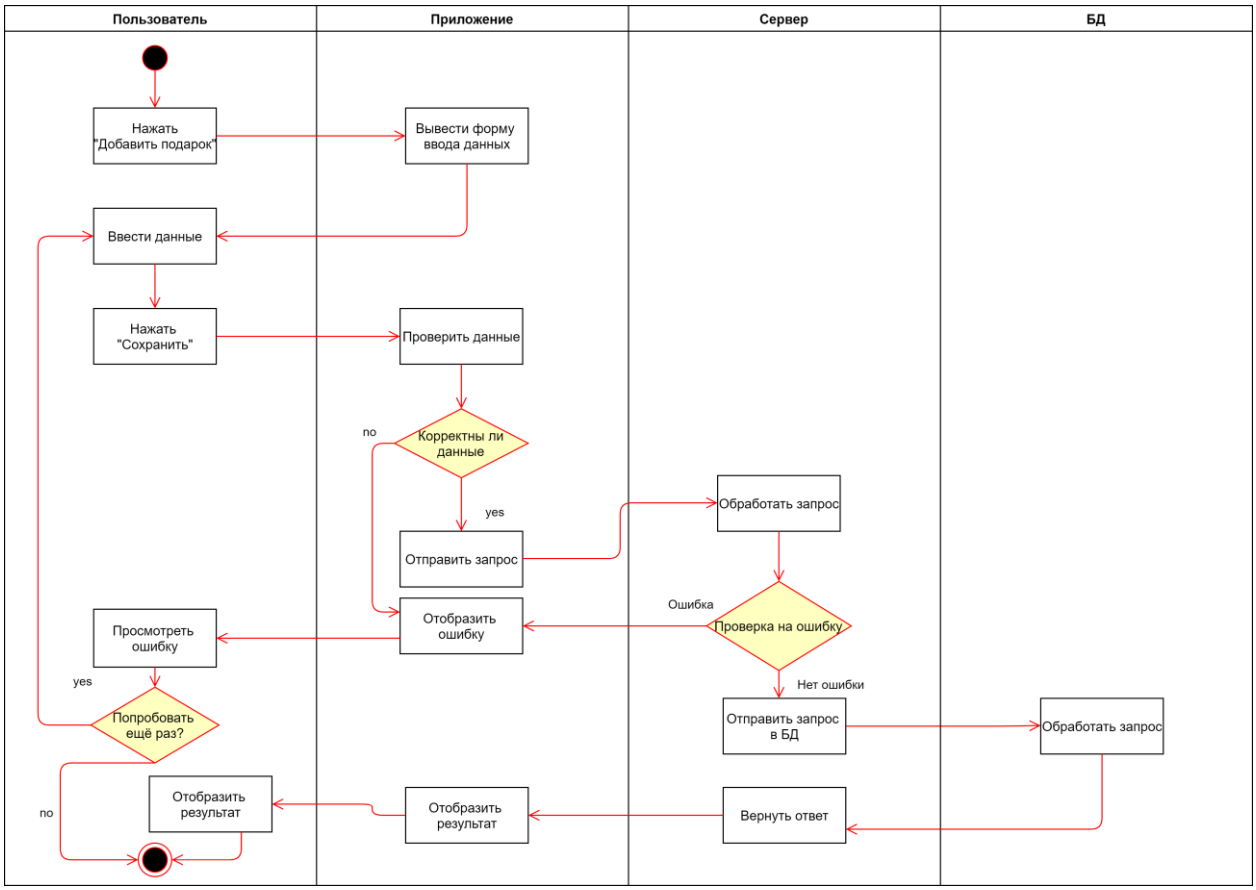

Рис. 15. Диаграмма активности для создания желания

Авторизованный пользователь может добавить новое желание с главного экрана, нажав кнопку «Добавить желание» (обозначена как + с подарком). Приложение отобразит окно для ввода данных. Пользователю необходимо ввести данные о новом желании и нажать кнопку «Сохранить».

Далее приложение проверяет введенные данные, если данные корректны, приложение отправляет запрос на сервер. Если пользователь ввел неверные данные, приложение выведет пользователю ошибку. После чего пользователь может заново ввести данные или закончить действие. После отправки запроса на сервер, сервер обрабатывает запрос и отправлять запрос к базе данных. После чего получает данные от БД и возвращает ответ. Приложение возвращает результат пользователю, и пользователь просматривает результат.

Если при обработке на сервере произошла ошибка или ответ от сервера не получен, приложение отображает пользователю ошибку, пользователь просматривает ошибку, после чего может попробовать еще раз или завершить действие.

2.3.8 Диаграмма классов

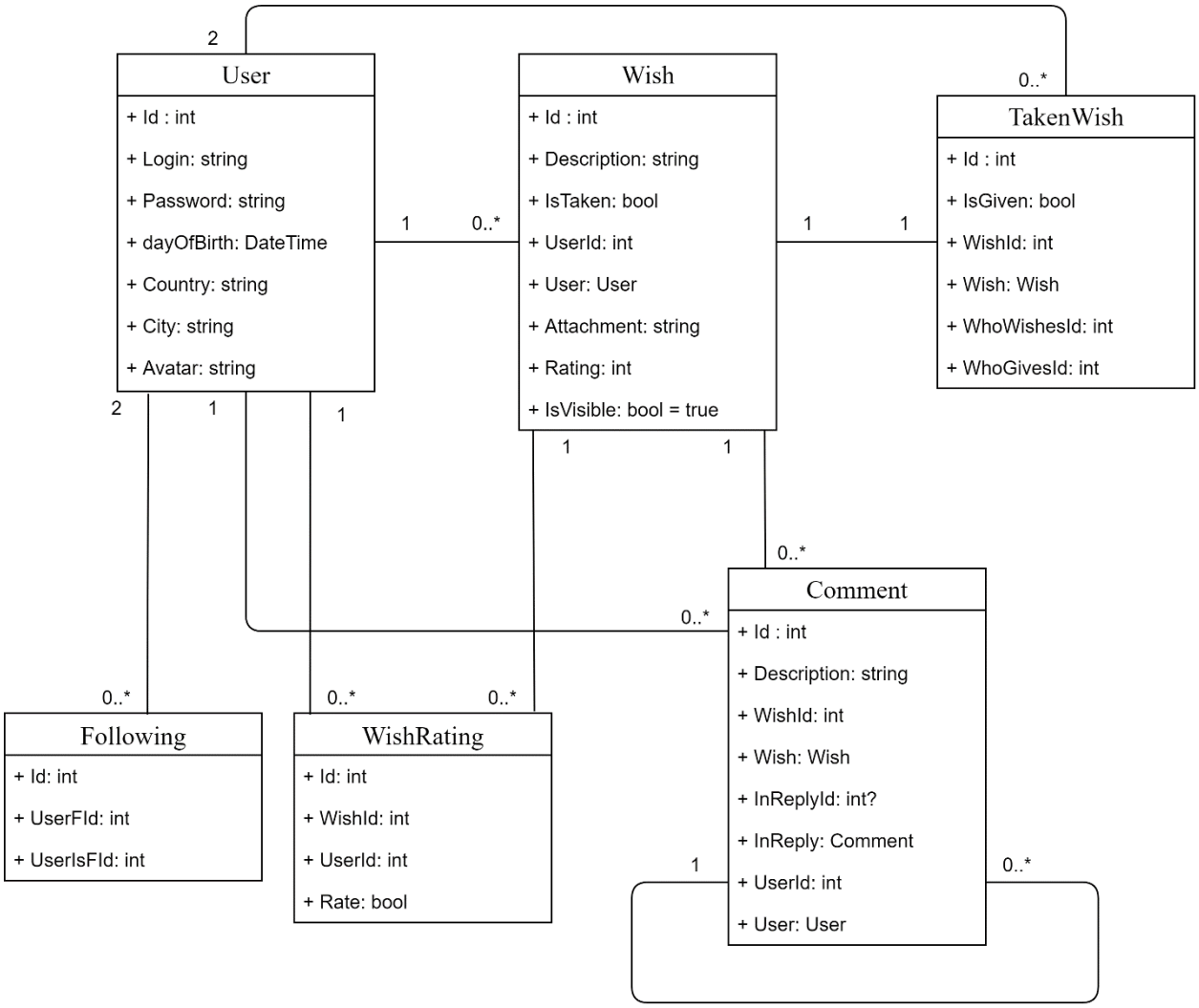

Рис. 16. Диаграмма классов

Имеются несколько классов:

User, обозначающий пользователя приложения:

id - уникальный идентификатор

login - имя пользователя в системе

password - пароль

dayOfBirth - дата рождения

Country - страна проживания

City - город

Avatar - аватар пользователя;

Wish, обозначающий желания пользователей:

id - уникальный идентификатор

description - описание желания

IsTaken - хочет ли какой-то другой пользователь подарить этот предмет

UserId - уникальный идентификатор автора данного желания

User - автор желания

Attachment - изображение желания

Rating - рейтинг данного желания

IsVisible - видно ли это желание в приложении. Если у желания низкий рейтинг, то оно скрывается. По умолчанию все желания видны;

Following, обозначающий подписки пользователя на других пользователей:

id - уникальный идентификатор

UserFId - пользователь, который подписывается

UserIsFId - пользователь, на которого подписываются;

WishRating, обозначающий рейтинг(+ или -) того или иного желания от конкретного пользователя:

id - уникальный идентификатор

WishId - уникальный идентификатор желания

UserId - пользователь, который оценивает желание

Rate - сама оценка(“+” это true, “-” это false). Сделано для того, чтобы один пользователь не мог несколько раз поставить отрицательную или положительную оценку желанию;

WishRating был введён с целью контроля контента, который публикуют пользователь. Если рейтинг желания достигнет определённого порога, то оно будет удалено в связи с тем, что по мнению других пользователей представляет неуместный контент.

Comment, обозначающий комментарий под тем или иным желанием:

Id - уникальный идентификатор

Description - сам комментарий

WishId - уникальный идентификатор желания, под которым оставляется комментарий

Wish - само желание

InReply - дописать

InReplyId - дописать

UserId - уникальный идентификатор автора комментария

User - автор комментария;

TakenWish создан для сохранения информации о том, какой пользователь поставил отметку «Подарить» определённому желанию другого пользователя:

Id - уникальный идентификатор

IsGiven – флаг о том, что подарок был вручён автору желания

WishId - уникальный идентификатор желания, которое было отмечено как «подарить»

Wish - само желание

WhoWishesId - уникальный идентификатор автора желания

WhoGivesId – уникальный идентификатор пользователя, который хочет подарить данный подарок.

2.4 Воронки в Яндекс Метрике.

2.4.1 Воронка “Создание желания”

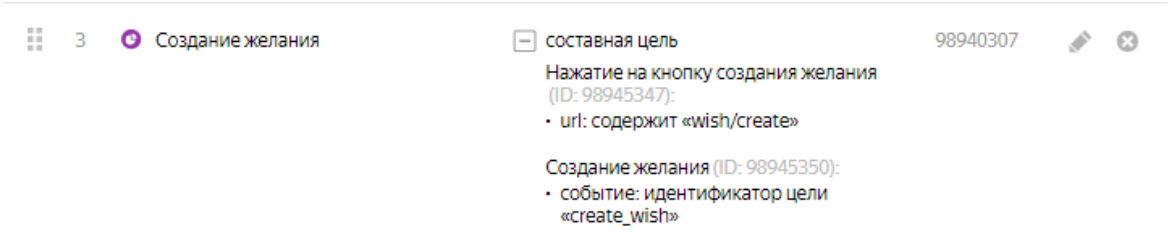

Рис. 17. Настройка воронки «Создание желания»

Рис. 18. Графическая иллюстрация воронки «Создание желания»

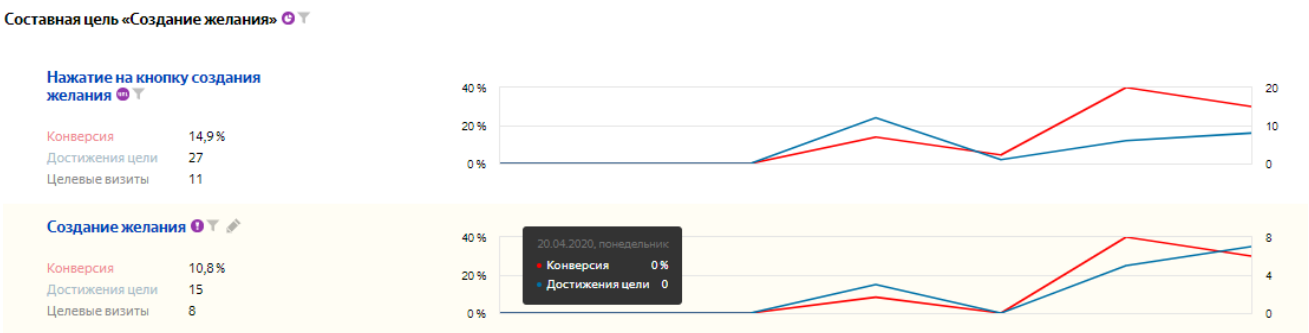

Рис. 19. Конверсия воронки «Создание желания»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “/wish/create”;
- нажать на кнопку “Сохранить”, которая вызывает выполнение кода, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.2 Воронка «Оценка желаний»

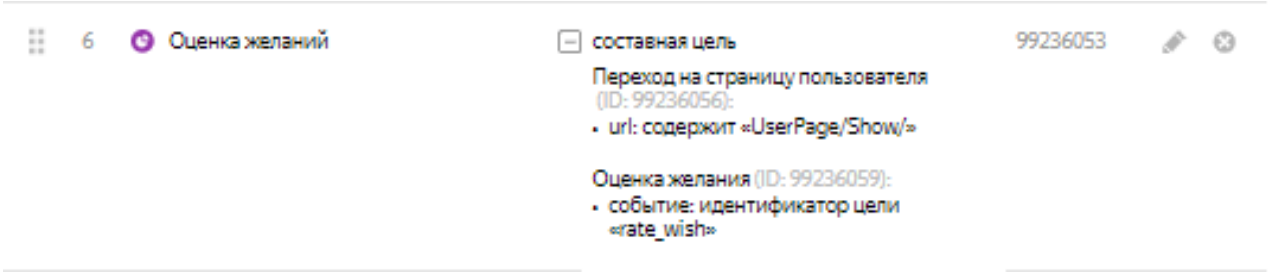

Рис. 20. Настройка воронки «Оценка желаний»

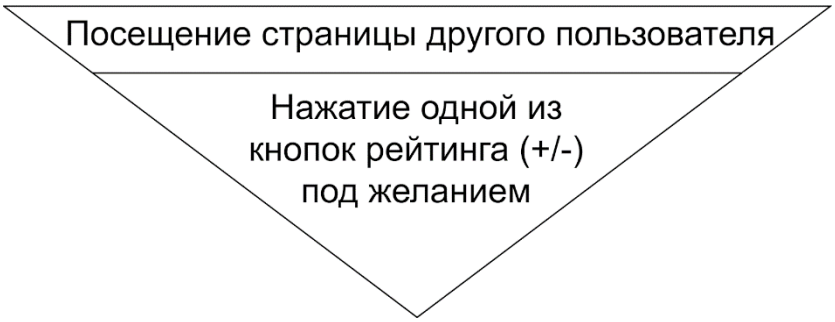

Рис. 21. Графическая иллюстрация воронки «Оценка желаний»

Рис. 22. Конверсия воронки «Оценка желаний»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “UserPage/Show/”;
- нажать на одну из кнопок (+ или -) расположенных рядом с каждым желанием (если текущая страница не является собственной страницей пользователя), которая вызывает выполнение кода, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.3 Воронка «Добавление комментария»

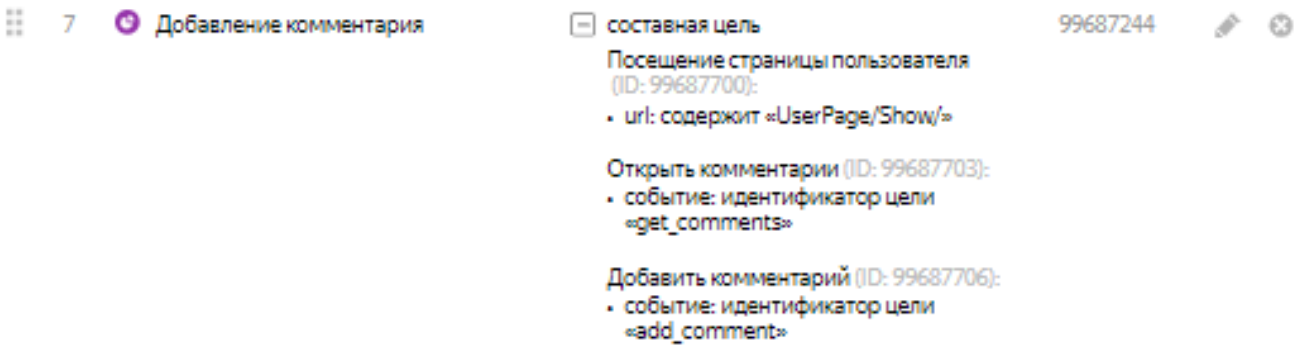

Рис. 23. Настройка воронки «Добавление комментария»

Рис. 24. Графическая иллюстрация воронки «Добавление комментария»

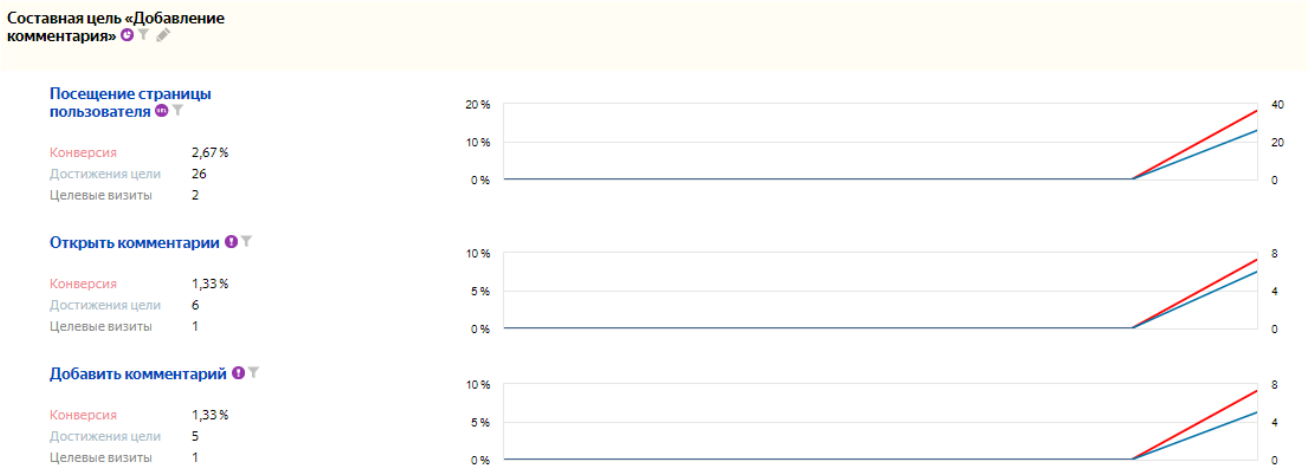

Рис. 25. Конверсия воронки «Добавить комментарий»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 3 действия:

- перейти на страничку в URL адресе которой содержится строка “UserPage/Show/”;
- нажать на одну из кнопок “Комментарии” расположенных рядом с каждым желанием, которая вызывает выполнение кода, отправляющего данные о достижении заданной цели в Яндекс Метрику;
- нажать на кнопку “Отправить”, которая вызывает выполнение кода, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.4 Воронка «Отметка “Подарить”»

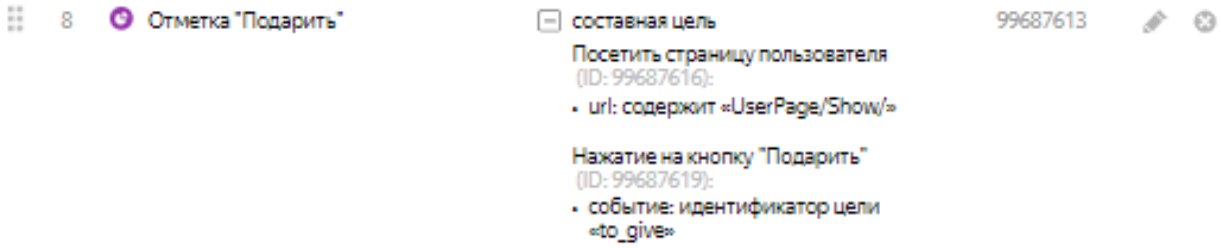

Рис. 26. Настройка воронки «Отметка “Подарить”»

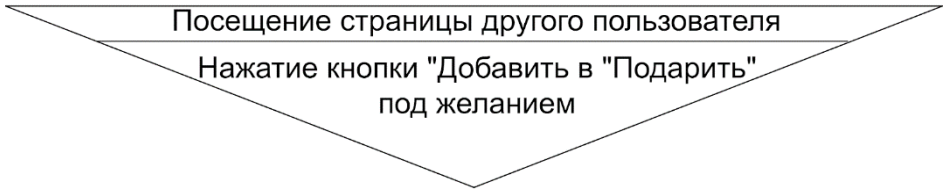

Рис. 27. Графическая иллюстрация воронки «Отметка “Подарить”»

Рис. 28. Конверсия воронки «Отметка “Подарить”»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “UserPage/Show/”;
- нажать на одну из кнопок “Добавить в “Подарить” расположенных рядом с каждым желанием (если текущая страница не является собственной страницей пользователя), которая вызывает выполнение кода, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.4.5 Воронка «Отметить как “Подарено”»

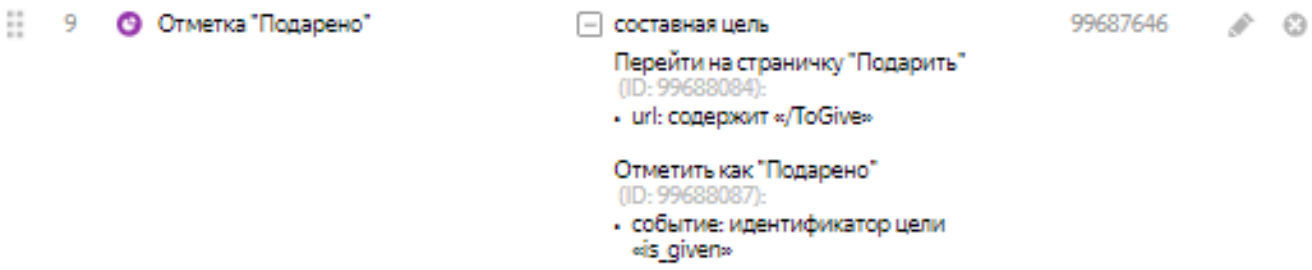

Рис. 29. Настройка воронки «Отметить как “Подарено”»

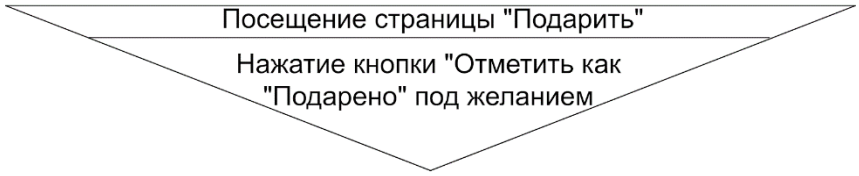

Рис. 30. Графическая иллюстрация воронки «Отметить как “Подарено”»

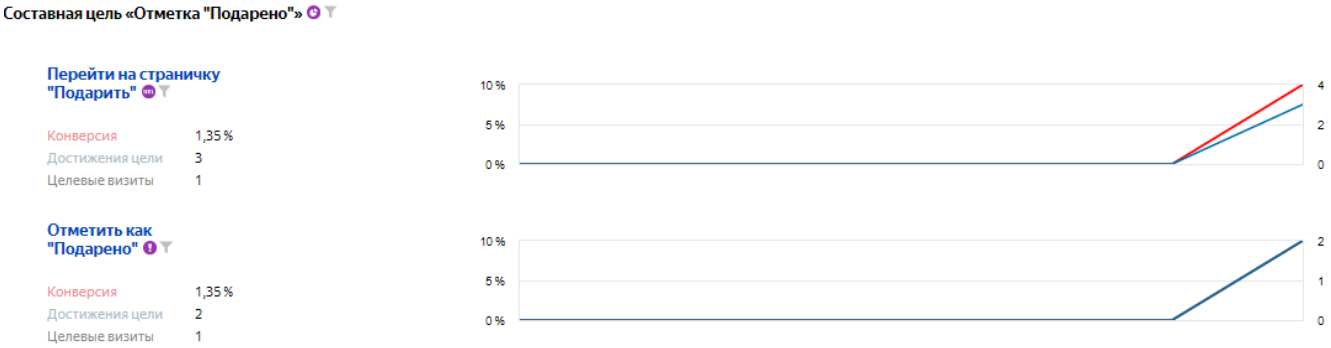

Рис. 31. Конверсия воронки «Отметить как “Подарено”»

Для выполнения условий данной воронки пользователь должен последовательно выполнить 2 действия:

- перейти на страничку в URL адресе которой содержится строка “/ToGive”;
- нажать на одну из кнопок “Отметить как “Подарено” расположенных рядом с каждым желанием, которая вызывает выполнение JS скрипта, отправляющего данные о достижении заданной цели в Яндекс Метрику.

2.5 Схема базы данных

На рисунке 32 представлена ER-диаграмма, на которой показаны основные сущности системы и их взаимосвязь между собой.

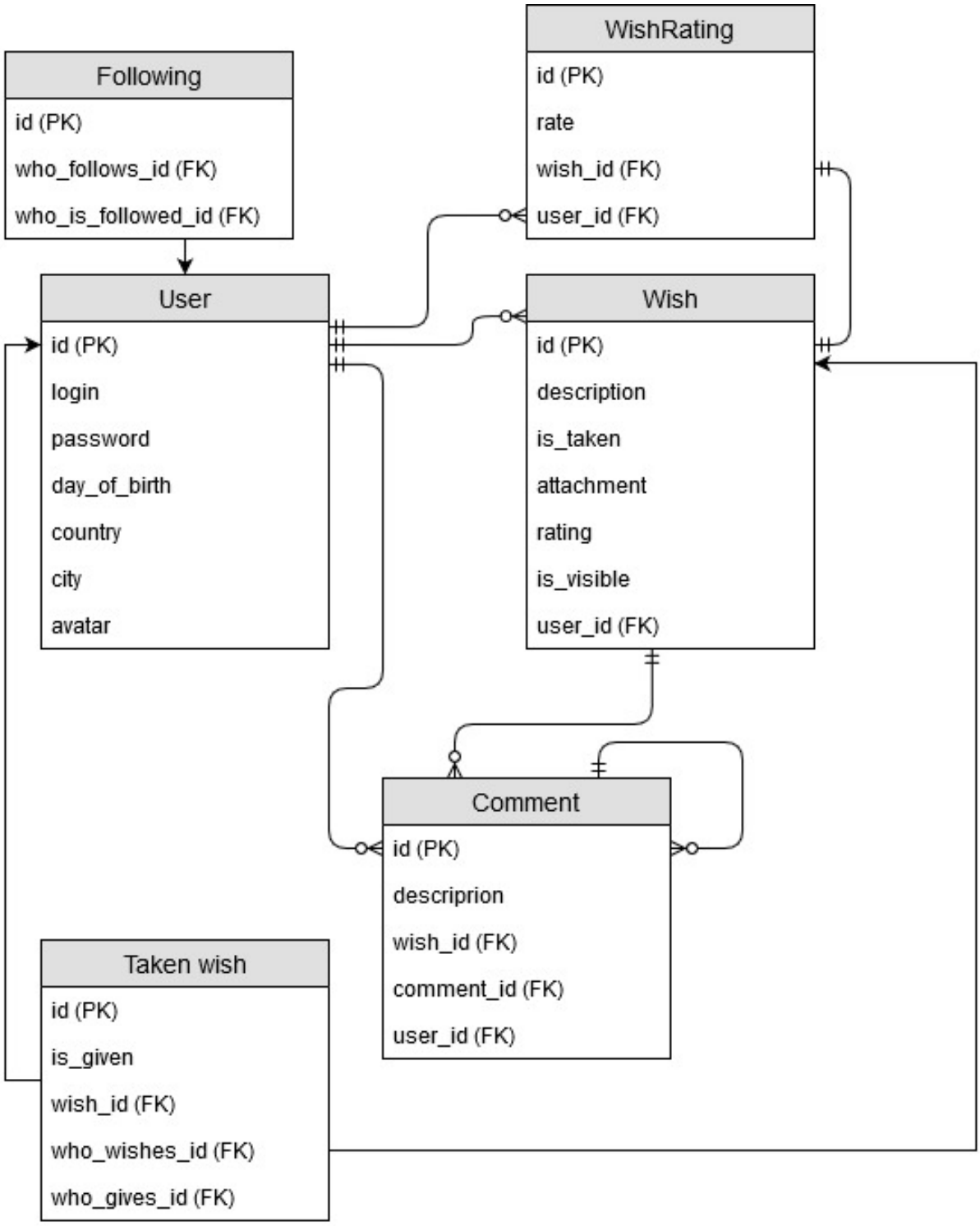

Рис. 32. ER-диаграмма

- У пользователя может быть одно или несколько желаний (или не быть их совсем), а желания обязательно должен быть только один пользователь. Т.к. связь 1:m, в сущности “Wish” содержится внешний ключ “user_id”, который будет указывать на id пользователя, который создал данное желание.
- Т.к. к желанию можно написать комментарий, получается, что у желания может быть один или несколько комментариев (или может не быть вовсе), а комментарий должен обязательно ссылаться только на одно желание и только на одного пользователя. В функциональности так же предусмотрен ответ на другой комментарий, поэтому будет существовать связь между сущностями “Comment” как 1:m, где у комментария может быть один или несколько ответных комментарий (или может не быть вовсе), а ответный комментарий обязательно должен ссылаться только на один комментарий.
- “Taken wish” – это соединяющая таблица для “User” и “Wish”, где хранятся данные о том, какое желание было отмечено, каким пользователем, кто создал данное желание и было ли оно подарено.
- “Following” – таблица, хранящая данные о том, какой пользователь на кого подписан.
- “WishRating” – таблица, хранящая данные о том, какой рейтинг (в большую сторону или в меньшую) поставил определённый пользователь определённому желанию другого пользователя. Конкретный рейтинг может относится только к одному желания, а пользователь может поставить рейтинг для многих желаний, а оценка пользователя относится только к этому конкретному пользователю.

3. Обоснование архитектуры проекта

3.1 Определение архитектуры проекта

В качестве архитектуры приложения был взят паттерн MVC - Model-View-Controller (рисунок 33).

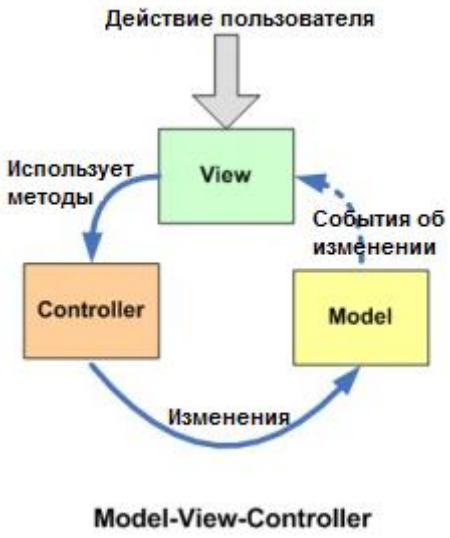

Рис. 33. Схема работы MVC

Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. [3]

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные

одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1) к одной модели можно присоединить несколько видов, при этом не затрагивая реализацию модели. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;

2) не затрагивая реализацию видов, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой контроллер;

3) ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (модели), вообще не будут осведомлены о том, какое представление будет использоваться.

3.2 Сравнение с другими паттернами

Рассмотрим различия между паттерном MVC и паттернами MVP, MVVM.

MVP - Model-View-Presenter (рисунок 34).

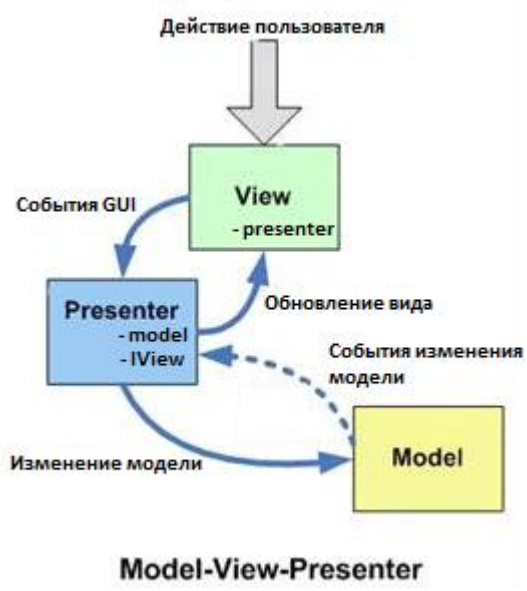

Рис. 34. Схема работы MVP

Данный подход позволяет создавать абстракцию представления. Для этого необходимо выделить интерфейс представления с определенным набором свойств и методов. Презентер, в свою очередь, получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель.

Признаки презентера:

- двухсторонняя коммуникация с представлением;
- представление взаимодействует напрямую с презентером, путем вызова соответствующих функций или событий экземпляра презентера;
- презентер взаимодействует с View путем использования специального интерфейса, реализованного представлением;
- один экземпляр презентера связан с одним отображением.

Реализация:

Каждое представление должно реализовывать соответствующий интерфейс. Интерфейс представления определяет набор функций и событий, необходимых для взаимодействия с пользователем (например, IView.ShowErrorMessage(string msg)). Презентер должен иметь ссылку на реализацию соответствующего интерфейса, которую обычно передают в конструкторе.

Логика представления должна иметь ссылку на экземпляр презентера. Все события представления передаются для обработки в презентер и практически никогда не обрабатываются логикой представления (в т.ч. создания других представлений)[2].

Пример использования: Windows Forms.

Правила применения:

- используется в ситуации, когда невозможно связывание данных (нельзя использовать Binding);
- частым примером может быть использование Windows Forms.

MVVM - Model-View-View Model (рисунок 35).

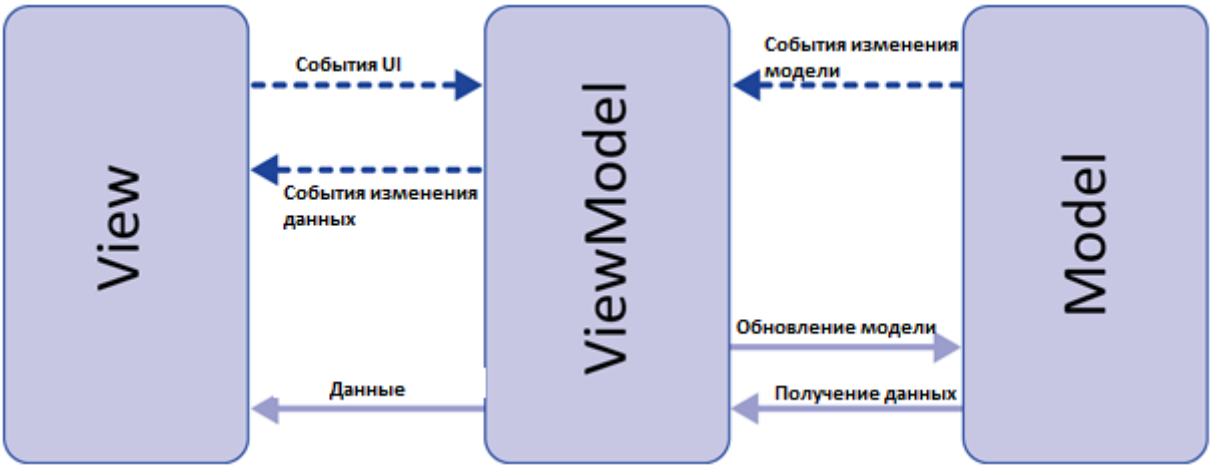

Рис. 35. Схема работы MVVM

Данный подход позволяет связывать элементы представления со свойствами и событиями View-модели. Можно утверждать, что каждый слой этого паттерна не знает о существовании другого слоя.

Признаки View-модели:

- двухсторонняя коммуникация с представлением;
- View-модель — это абстракция представления. Обычно означает, что свойства представления совпадают со свойствами View-модели / модели;
- View-модель не имеет ссылки на интерфейс представления (IView). Изменение состояния View-модели автоматически изменяет представление и наоборот, поскольку используется механизм связывания данных (Bindings);
- один экземпляр View-модели связан с одним отображением.

Реализация:

При использовании этого паттерна, представление не реализует соответствующий интерфейс (IView).

Представление должно иметь ссылку на источник данных (DataContex), которым в данном случае является View-модель. Элементы представления связаны (Bind) с соответствующими свойствами и событиями View-модели.

В свою очередь, View-модель реализует специальный интерфейс, который используется для автоматического обновления элементов представления. Примером такого интерфейса в WPF может быть INotifyPropertyChanged.

Пример использования: WPF

Правила применения:

- используется в ситуации, когда возможно связывание данных без необходимости ввода специальных интерфейсов представления (т.е. отсутствует необходимость реализовывать IView);
- частым примером является технология WPF.

4. Требования к разрабатываемой системе

1) Приложение должно иметь клиент-серверную архитектуру.
2) Данные пользователя хранятся в базе данных на сервере.
3) При запросе данных, клиентское приложение должно отправлять соответствующий запрос на сервер, а сервер должен обрабатывать запрос и делать запрос к БД, после чего отправлять ответ.
4) При добавлении, изменении или удалении данных клиентское приложение должно отправлять соответствующий запрос на сервер, сервер должен обрабатывать запрос и делать запрос к БД, после чего отправлять ответ.
5) Общение клиента и сервера производится с помощью REST API запросов по HTTP протоколу.
6) Система должна выполнять обозначенные выше задачи, а именно:
    - создавать/редактировать/удалять своё «желание»;
    - отмечать подарок другого пользователя, который они хотят ему подарить, и удалять данную отметку;
    - подписываться через поиск по пользователям на другие профили и отписываться от получения новостей о добавлении новых желаний определённого пользователя;
    - оставлять комментарии к записи с желанием и удалять его;
    - изменять данные своего профиля, включая смену пароля.

5. Планирование работ

5.1 Виды работ, которые необходимо выполнить в процессе разработки программного средства

Разработка приложения должна быть проведена в три этапа:

1) разработка документации;
2) разработка приложения;
3) тестирование готового продукта.

На этапе разработки документации должны быть выполнены следующие работы:

1) написание технического задание согласно ГОСТ 34;
2) написание анализа предметной области, включающие:
   - актуальность выбранной задачи;
   - анализ существующих решений;
   - uml-диаграммы (диаграммы вариантов использования, последовательности, состояний, развертывания, объектов, активности);
   - схему базы данных.

На этапе разработки приложения должны быть выполнены следующие работы:

1. написание клиентского приложение, включающее:
   1.1. создание экранов, требуемых для осуществления всех функциональных требований:
      1.1.1. экран для просмотра страницы пользователя;
      1.1.2. экран для авторизации;
      1.1.3. экран для просмотра желаний;
      1.1.4. экран для просмотра подписчиков пользователя.
   1.2. создание панели управления, осуществляющей возможность перехода между экранами в п. 1.1;
   1.3. регистрация и авторизация в приложении;
   1.4. редактирование информации о пользователе;
   1.5. добавление желаний;
   1.6. редактирование желаний;
   1.7. удаление желаний;

1.8. оценка желаний;
1.9. добавление желаний другого пользователя в список «Подарить»;
1.10. удаление желаний другого пользователя из списка «Подарить»;
1.11. добавление другого пользователя в список «Подписчики»;
1.12. удаление пользователя из списка «Подписчики»;
1.13. добавление комментария к желанию;
1.14. удаление комментария к желанию.

2. написание серверной части, включающее:
   2.1. создание базы данных на сервере;
   2.2. написание REST-запросов для осуществления функциональных требований из п.1;
   2.3. размещение серверной части приложение на облачном хостинге;
   2.4. осуществление взаимодействия между клиентским приложением и сервером.

На этапе тестирования продукта должны быть выполнены следующие работы:

1. написание тест-плана;
2. составление тестов-кейсов;
3. проведение тестирования;
4. документирование результатов тестирования.

5.2 Состав команды, распределение задач по участникам

В состав команды разработчиков вошли:

1. Агафонова Марина;
2. Аленичев Александр;
3. Семечев Данила.

С таблицами по распределению задач по участникам можно ознакомиться в Приложении 1.

6. Реализация приложения

6.1 Анализ средств реализации

В качестве реализации клиент-серверного приложения были выбраны:

- Серверная часть была разработана на платформе .NET Core, поддерживающей язык программирования C#. Основным преимуществом .NET Core над многими платформами является кроссплатформенность, что позволяет запускать приложение на разных операционных системах;
- Клиентская часть сделана так же на .NET Core с помощью представлений cshtml, которые содержат пользовательский интерфейс, а также технологий HTML5, CSS, Bootstrap 4. Основные преимущества:
  - ускорение разработки за счёт готовой разметки;
  - кросс-браузерность.
- В качестве СУБД была выбрана MS SQL Server, поскольку эта система обладает высокой производительностью, интеллектуальной обработкой запросов. Также с этой системой работает решение Entity Framework Core на платформе .NET Core. EF Core может использоваться как объектно-реляционный модуль сопоставления (ORM), позволяя разработчикам .NET работать с базой данных с помощью объектов .NET и устраняя необходимость в написании большей части кода, требуемого для доступа к данным[1];
- Общение с сервером происходит посредством REST API по протоколу HTTP запросами и ответами в формате JSON, для обработки запросов также используется технология AJAX.

6.2 Выполнение требований по безопасности

В целях повышения безопасности использовалось расширение протокола HTTP - HTTPS (HyperText Transfer Protocol Secure), поддерживающее шифрование. Для хеширования паролей применялся алгоритм MD5. (сертификат?)

6.3 Разработка API

Для создания и описания REST API было использовано решение Microsoft.AspNet.WebApi.Core для платформы .NET Core, что позволяет быстро и удобно настроить API.

Для документации REST API была использована технология Swagger. Ниже представлен интерфейс, описывающий API системы.

Рис. 36. Использование Swagger для описания API системы (часть 1)

Рис. 37. Использование Swagger для описания API системы (часть 2)

Рис. 38. Использование Swagger для описания API системы (часть 3)

7. Тестирование

После реализации всех задач, был проведен запланированный набор тестов. Он включает 3 вида тестирования:

- дымовое тестирование;
- модульное тестирование;
- интеграционное тестирование.

7.1 Дымовое тестирование (smoke testing)

Цель тестирования – проверка готовности разработанного продукта к проведению более расширенного тестирования.

Для данного тестирования необходимо было проверить работоспособность сайта на следующих основных сценариях:

- регистрация;
- авторизация;
- создание подарка;
- редактирование подарка;
- удаление подарка;
- отметка подарка в «Подарить»;
- удаление отметки у подарка в «Подарить»;
- отметка у подарка как «Подарено»;
- изменение рейтинга подарка;
- добавление комментария к подарку;
- удаление комментария у подарка;
- переход на страницу профиля пользователя;
- редактирование пользователя;
- смена пароля у пользователя;
- поиск пользователей;
- подписка на пользователя;
- удаление из подписок пользователя.

Дымовое тестирование проводилось вручную, в следующих браузерах: Mozilla Firefox, Google Chrome, Opera с включенным WI-FI для связи с back-end частью сайта. Результаты, полученные в ходе тестирования представлены в таблице 1.

Таблица 1. Результаты дымового тестирования

Сценарий	**Результат**
Регистрация	Пройдено
Авторизация	Пройдено
Создание подарка	Пройдено
Редактирование подарка	Пройдено
Удаление подарка	Пройдено
Отметка подарка в «Подарить»	Пройдено
Удаление отметки у подарка в «Подарить»	Пройдено
Отметка у подарка как «Подарено»	Пройдено
Изменение рейтинга подарка	Пройдено
Добавление комментария к подарку	Пройдено
Удаление комментария у подарка	Пройдено
Переход на страницу профиля пользователя	Пройдено
Редактирование пользователя	Пройдено
Смена пароля у пользователя	Пройдено
Поиск пользователей	Пройдено
Подписка на пользователя	Пройдено

Удаление из подписок пользователя	Пройдено

По итогу тестирования выяснилось, что сайт проходит все основные сценарии.

7.2 Модульное тестирование (unit testing)

В рамках данного тестирования проверялась отдельная часть логики приложения. Для данного тестирования были взяты следующие сценарии:

1) редактирование аккаунта авторизованного пользователя;
2) удаление комментария, который принадлежал авторизованному пользователю;
3) подписка на пользователя;
4) редактирование желания авторизованного пользователя.

Модульное тестирование проводилось с помощью инструмента для тестирования xUnit.net.

Таблица 2. Результаты модульного тестирования

Сценарий	**Результат**
Редактирование аккаунта	Пройден
Удаление комментария	Пройден
Подписка на пользователя	Пройден
Редактирование желания	Пройден

7.3 Интеграционное тестирование (Integration testing).

В рамках данного тестирования проверялась работа компонентов приложения на уровне, который включает инфраструктуру, поддерживающую приложение (база данных, файловая система). Для данного тестирования были взяты следующие сценарии:

1) загрузка главной страницы;
2) отображение списка подписок;
3) отображение списка желаний авторизованного пользователя;
4) отображение страницы пользователя с определённым идентификатором (для теста был взял Id=1).

Интеграционное тестирование проводилось с помощью инструмента для тестирования xUnit.net.

Таблица 3. Результаты интеграционное тестирования

Сценарий	**Результат**
Загрузка главной страницы	Пройден
Отображение списка подписок	Пройден
Отображение списка желаний	Пройден
Отображение страницы пользователя	Пройден

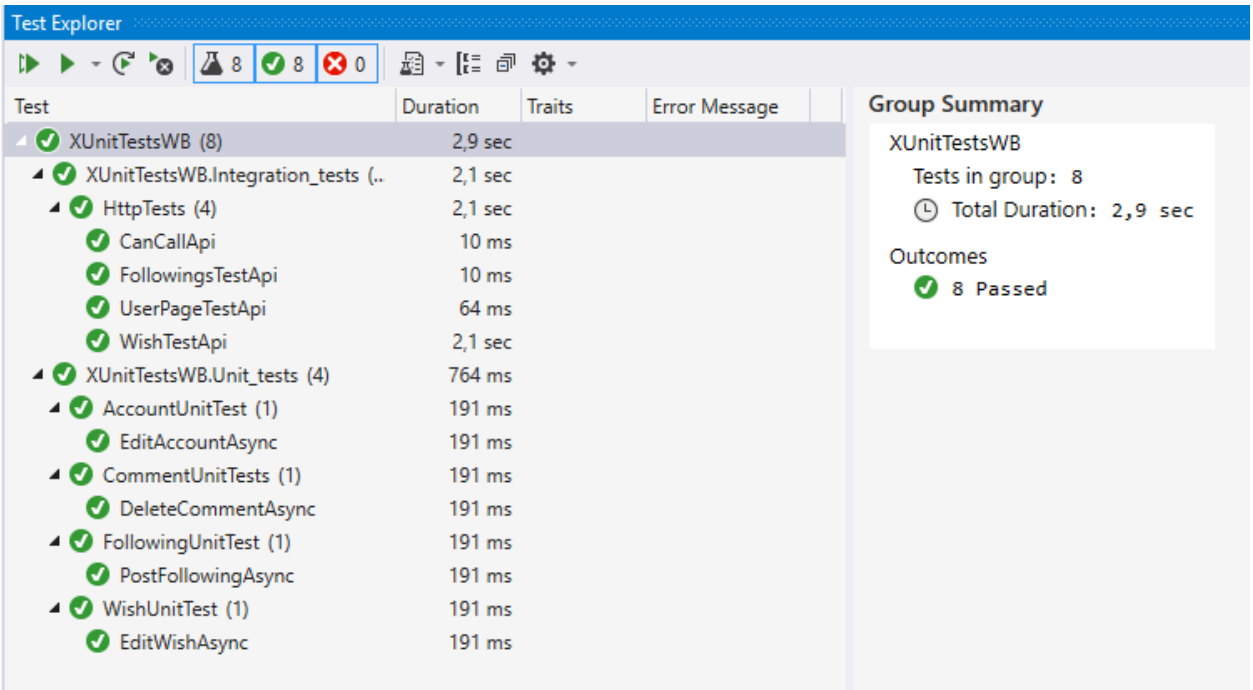

Рис. 39. Результаты тестирования с помощью xUnit.net

8. Интерфейс приложения

8.1 Главная страница авторизованного пользователя

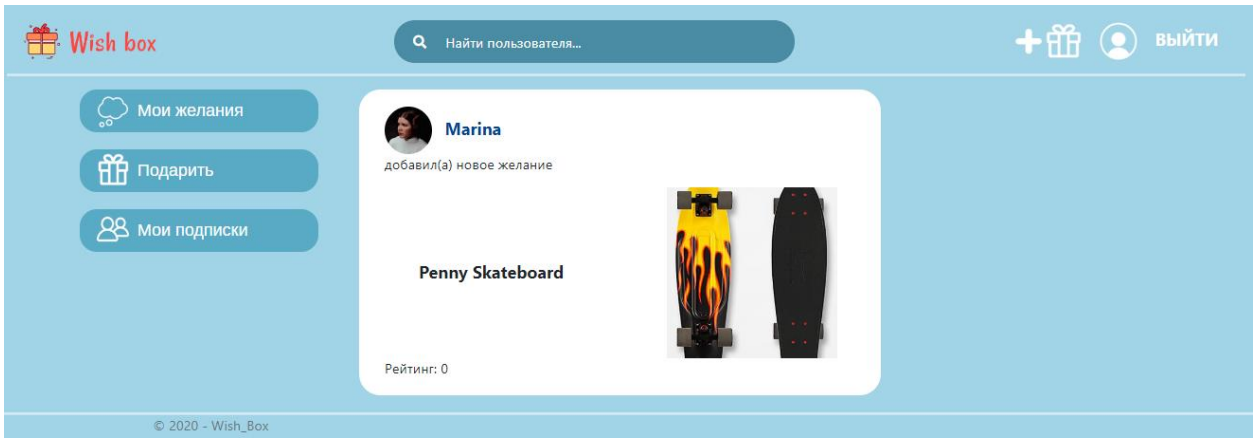

Рис. 40. Главная страница авторизованного пользователя

На данной странице авторизованный пользователь будет видеть новости других пользователей, на которых он подписался.

Пользователь может:

- перейти на страницу создания желания;
- перейти на страницу своего профиля;
- выйти из аккаунта;
- перейти на страницу своих желаний;
- перейти на страницу со списком того, что он желает подарить;
- перейти на страницу своих подписок;
- произвести поиск по имени.

8.2 Страница «Мои желания»

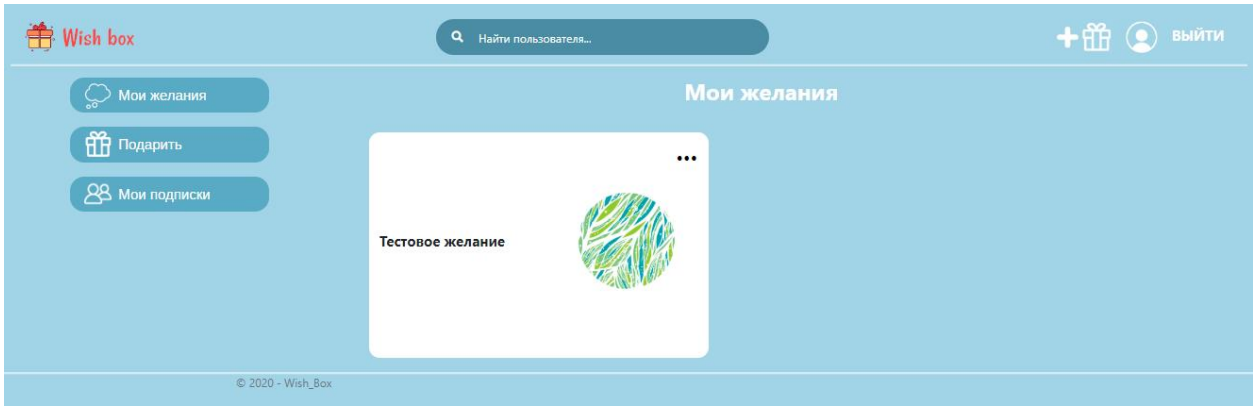

Рис. 41. Страница «Мои желания»

Данная страница доступна только авторизованным пользователям. На ней отображается список всех желаний авторизованного пользователя.

Пользователь может:

- всё то же, что и на главной странице;
- удалять или изменять свои желания при помощи меню, открывающегося кнопкой в виде трёх точек.

8.3 Страница «Подарить»

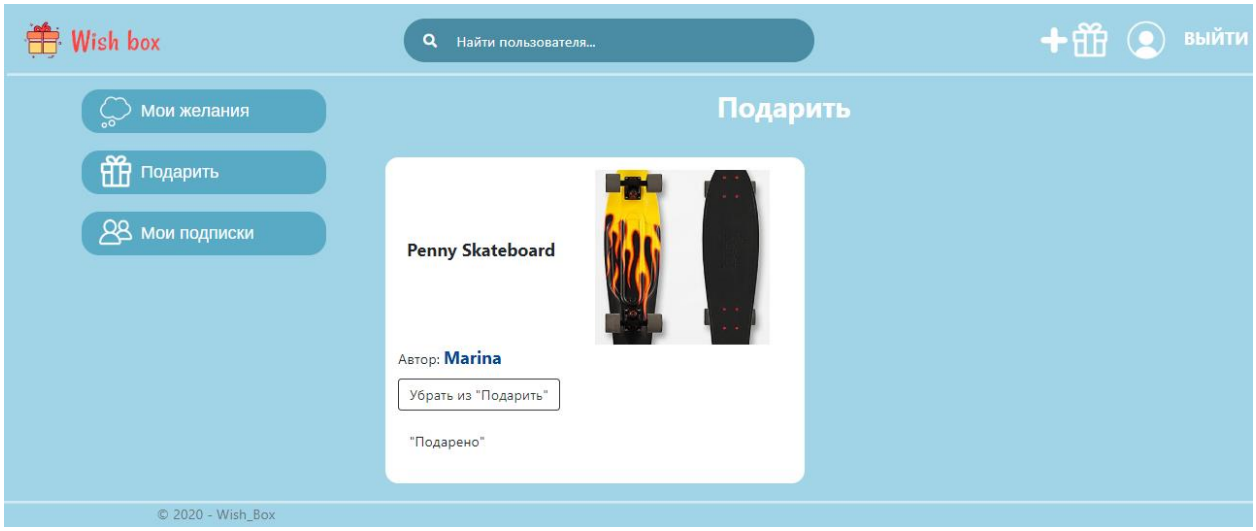

Рис. 42. Страница «Подарить»

Данная страница доступна только авторизованным пользователям. На ней отображается список всех желаний других пользователей, которые авторизованный пользователь добавил в свой список «Подарить».

Пользователь может:

- всё то же, что и на главной странице;
- отметить любой подарок, как подаренный;
- убрать из своего списка «Подарить» (отказаться дарить);
- зайти на страничку пользователя, являющегося автором желания.

8.4 Страница «Мои подписки»

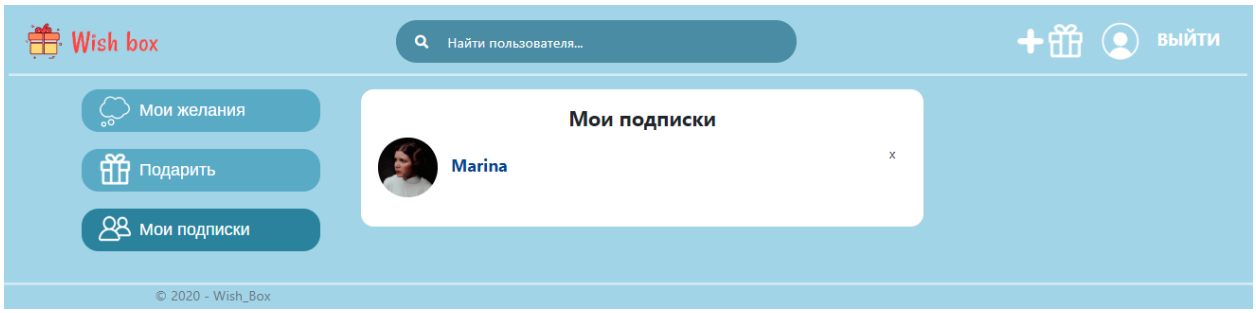

Рис. 43. Страница «Мои подписки»

Данная страница доступна только авторизованным пользователям. На ней отображается список всех подписок авторизованного пользователя.

Пользователь может:

- всё то же, что и на главной странице;
- удалять подписки при помощи кнопки в виде крестика;
- переходить на страничку пользователя, на которого есть подписка.

8.5 Страница добавления желания

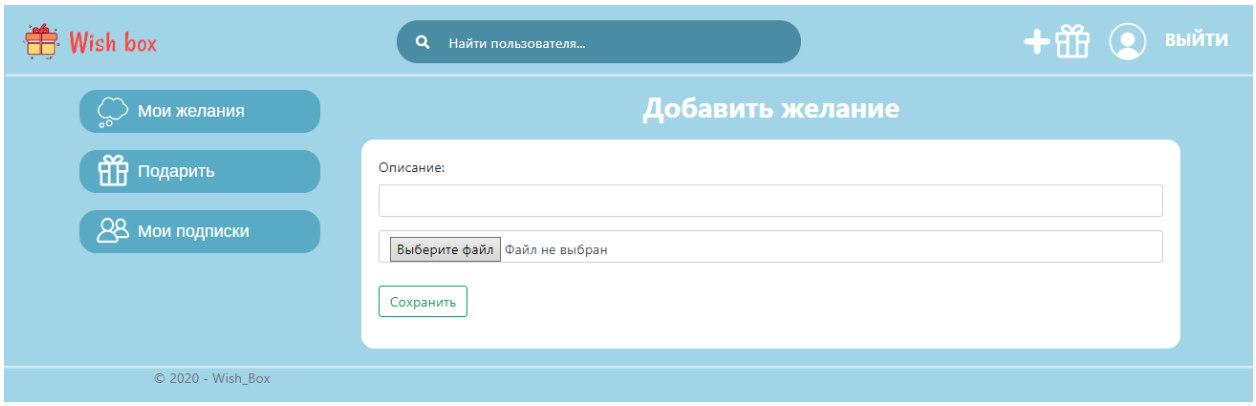

Рис. 44. Страница добавления желания

Данная страница доступна только авторизованным пользователям.

Пользователь может:

- всё то же, что и на главной странице;
- ввести необходимые данные для нового желания и нажать кнопку «Сохранить», для добавления желания.

8.6 Страница результатов поиска

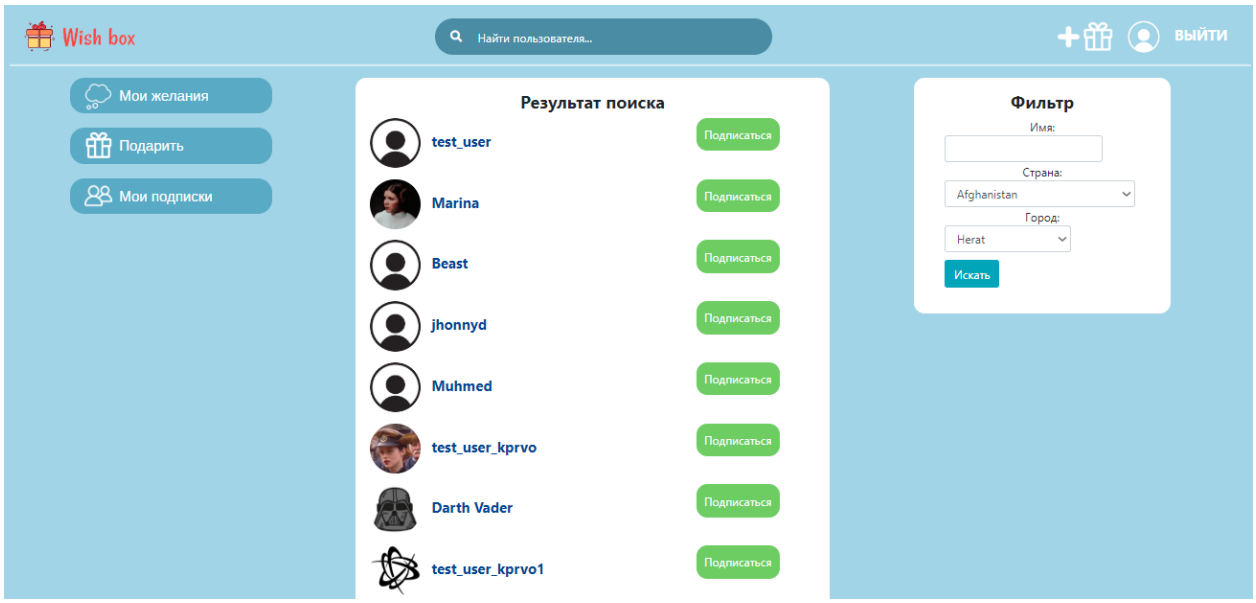

Рис. 45. Страница результатов поиска

Данная страница доступна только авторизованным пользователям. Она отображает результат поиска с заданными фильтрами, которые можно настроить в правой части экрана.

Пользователь может:

- всё то же, что и на главной странице;
- перейти на страницу другого пользователя;
- подписаться на пользователя;
- отписаться от пользователя.

8.7 Страница профиля авторизованного пользователя

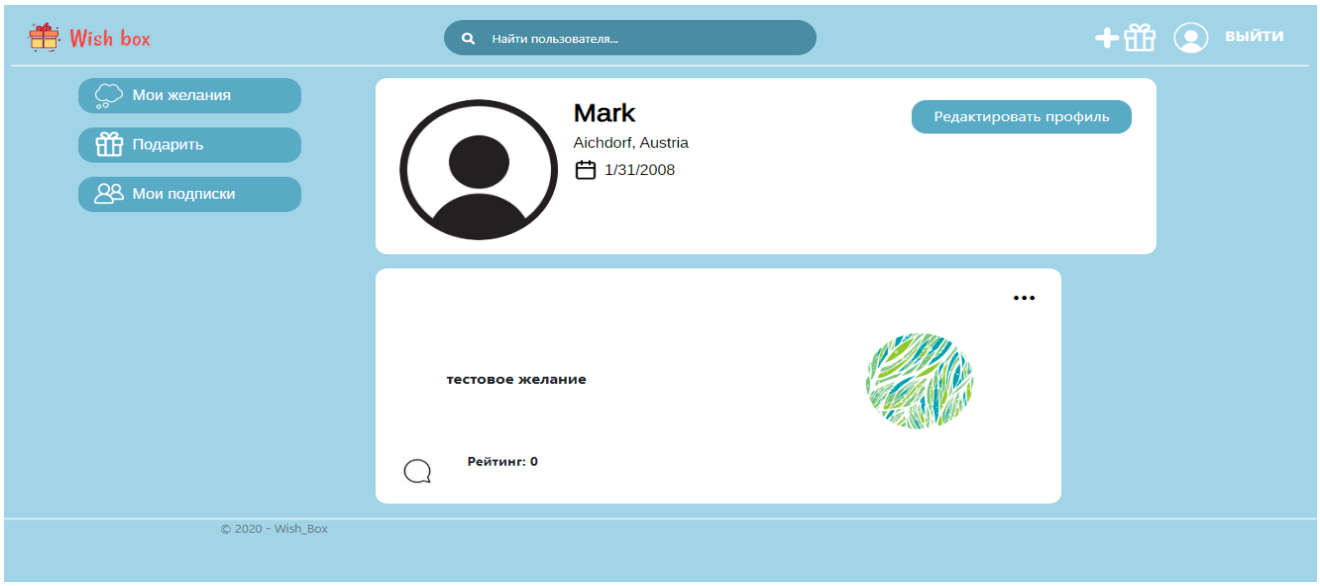

Рис. 46. Страница профиля авторизованного пользователя

Данная страница доступна только авторизованным пользователям.

Пользователь может:

- всё то же, что и на главной странице;
- показать или скрыть комментарии под своими желаниями;
- удалять или изменять свои желания;
- перейти на страницу редактирования своего профиля.

8.8 Страница редактирования профиля

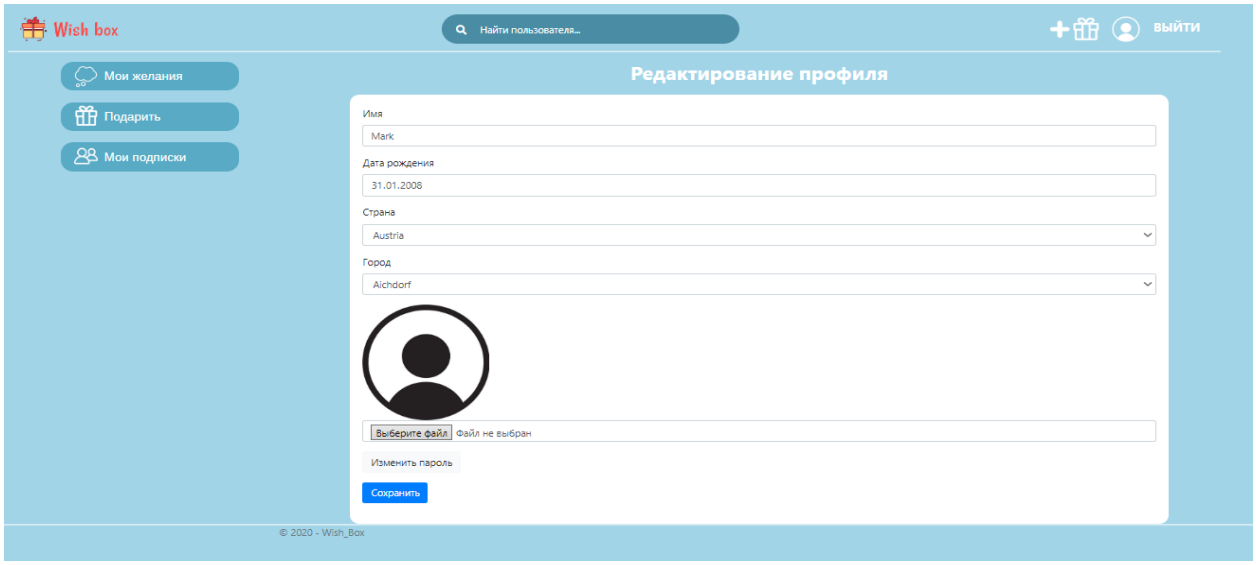

Рис. 47. Страница редактирования профиля

Данная страница доступна только авторизованным пользователям.

Пользователь может:

- всё то же, что и на главной странице;
- внести изменения в свой профиль и сохранить их нажатием на соответствующую кнопку;
- перейти на страницу изменения пароля

8.9 Страница изменения пароля

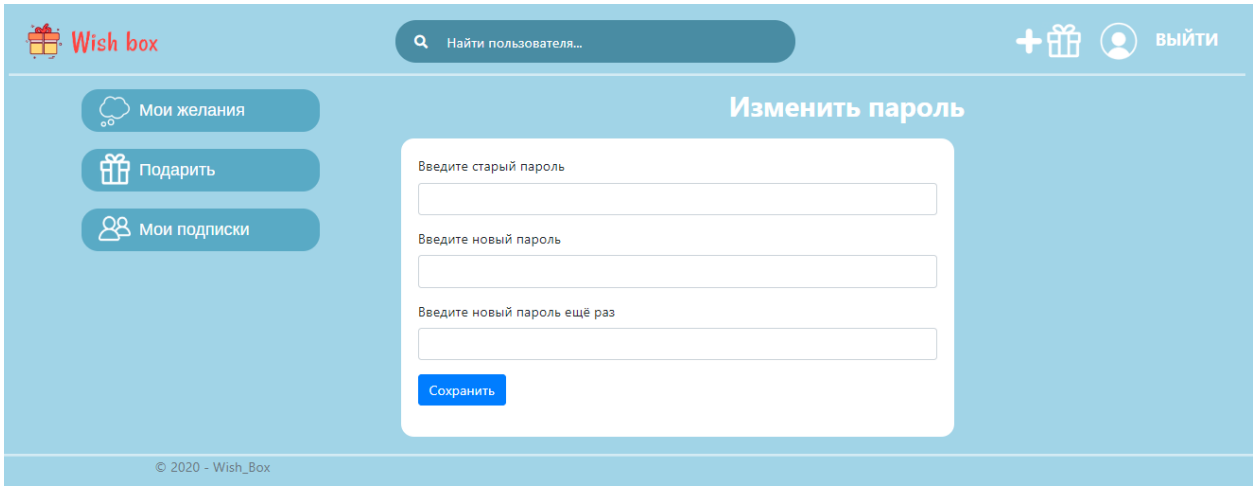

Рис. 48. Страница изменения пароля

Данная страница доступна только авторизованным пользователям.

Пользователь может:

- всё то же, что и на главной странице;
- изменить пароль для авторизации на сайте.

8.10 Страница другого пользователя

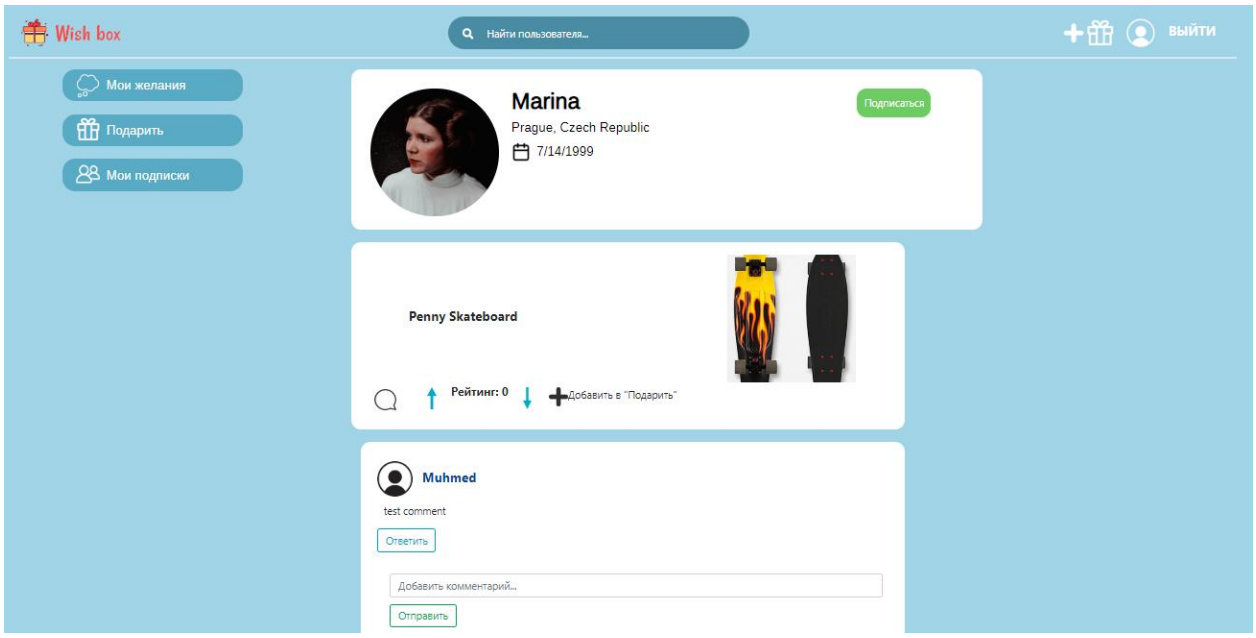

Рис. 49. Страница другого пользователя

Данная страница доступна только авторизованным пользователям.

Пользователь может:

- всё то же, что и на главной странице;
- подписаться на или отписаться от пользователя, на чьей страничке находится авторизованный пользователь;
- показать или скрыть комментарии под каждым желанием пользователя;
- оставить или удалить свой комментарий;
- поставить рейтинг каждому желанию;
- добавить в или убрать из списка желание пользователя.

8.11 Главная страница неавторизованного пользователя

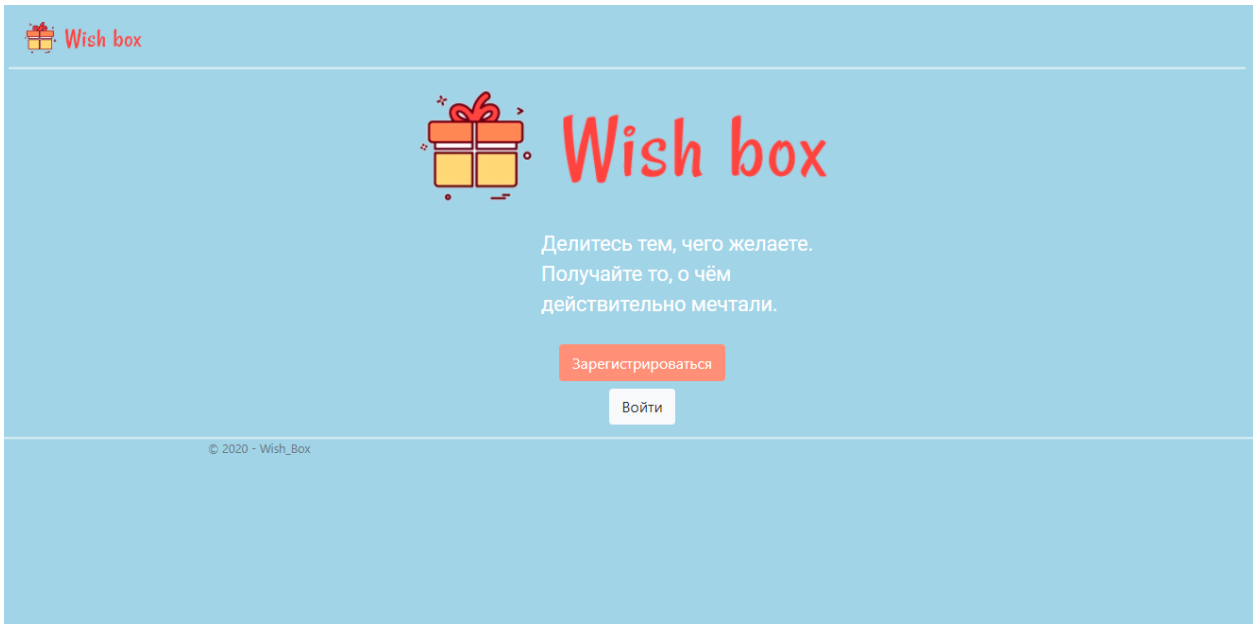

Рис. 50. Главная страница неавторизованного пользователя

Данная страница отображается всем неавторизованным посетителям, где они могут открыть форму регистрации или авторизации.

8.12 Форма регистрации

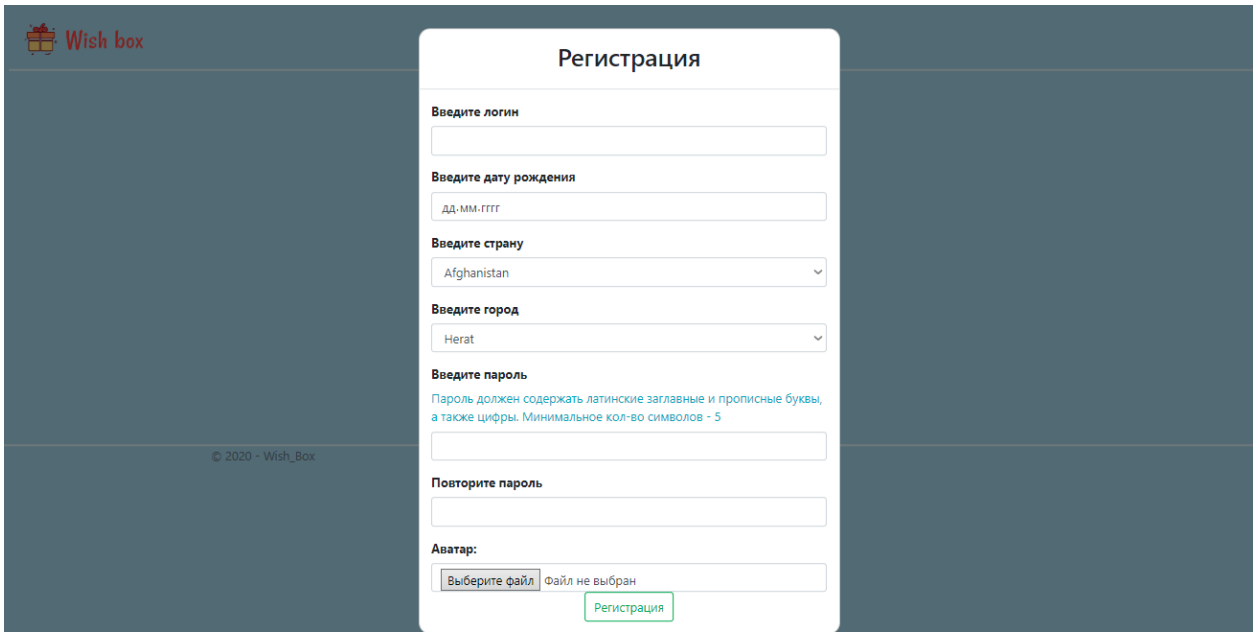

Рис. 51. Форма регистрации

Данная форма используется для регистрации новых пользователей. Все поля кроме поля «Аватар» являются обязательными для заполнения для успешной регистрации.

8.13 Форма авторизации

Рис. 51. Форма авторизации

Данная форма нужна для входа (авторизации) на сайте. Поля логин и пароль являются обязательными для ввода.

Заключение

В результате работы было сделано приложение для составления списка подарков и их отметки. Были реализованы следующие задачи:

- возможность создавать, редактировать и удалять подарки;
- возможность оставлять комментарии на подарке, а также удалять его;
- возможность отмечать подарок другого пользователя, который они хотят ему подарить, и удалять данную отметку;
- подписываться через поиск по пользователям на другие профили и отписываться от получения новостей о добавлении новых желаний определённого пользователя;
- возможность изменять данные своего профиля, включая смену пароля.

Приложение отвечает всем заявленным требованиям.

Список используемых источников

1. Entity Framework Core [Электронный ресурс] - URL: https://docs.microsoft.com/ru-ru/ef/core/ (дата обращения 09.06.20)
2. Model-View-Presenter — компромисс и универсальный рецепт [Электронный ресурс] – URL: https://habr.com/ru/post/343438/ (дата обращения 26.04.20)
3. MVC - Model View Controller (Модель-Вид-Контроллер) [Электронный ресурс] – URL: http://design-pattern.ru/patterns/mvc.html (дата обращения 27.04.20)
4. Чамберс Д. ASP.NET Core. Разработка приложений / Д. Чамберс, Д. Пэкетт. – П.: Питер, 2018. – 464 с.
5. Смит С. Построение современных веб-приложений с ASP.NET Core и Microsoft Azure / С. Смит, М. Вензел. – Redmond, Вашингтон, 2020. 101 с.

Приложение 1

Распределение задач по участникам команды

Таблица 4. – Распределение задач для Агафоновой Марины

№ п/п	Задача	Ссылка на GitHub
1.	Настройка проекта и подключение swagger	Настройка проекта, подключение swagger
2.	Классы моделей	Ссылка
3.	Работа с желанием	Ссылка
4.	Главная страница	Ссылка
5.	Front-end: Layout; Авторизация и Регистрация; Страницы для работы с желаниями; поиск и главная страница	Layout и авторизация, регистрация, поиск и главная страница
6.	Загрузка проекта в облако Azure	-
7.	Диаграммы: Use-Case; Последовательности; Состояния	-
8.	Видео	-
9.	Настройка REST API: Home; Wish; ToGive; Comments;	ссылка
10.	Тестирование: Unit; Integration	Unit; Integration

11.	Курсовая работа	-
12.	Классы – репозитории: Comments; Wish; WishRating; TakenWish; IRepository	TakenWish; Comments; Wish + WishRating; IRepository;

Распределение задач по участникам команды

Таблица 5. – Распределение задач для Аленичева Александра

№ п/п	Задача	Ссылка на GitHub
1.	Работа с комментарием	Ссылка
2.	Работа с желанием	Ссылка
3.	Список тех подарков, которые взял на себя текущий пользователь	Ссылка
4.	Front-end: Страница изменения пароля пользователя; Форма для добавление комментариев; список подарков, взятых текущим пользователем	Страница изменения пароля, форма для добавления комментариев, список подарков, взятых текущим пользователем
5.	Подключение аналитики	-
6.	Воронки: «Регистрация»; «Создание желания»; «Подписаться»; «Добавить комментарий»; «Отметка “Подарить”»; «Отметить как “Подарено”»	-

7.	Диаграммы: Развёртывания; Объектов	-
8.	Курсовая работа	-

Распределение задач по участникам команды

Таблица 6. – Распределение задач для Семечева Данилы

№ п/п	Задача	Ссылка на GitHub
1.	Система авторизации и регистрации	Ссылка
2.	Рейтинг желаний пользователя + модель данных WishRating	Ссылка
3.	Страница профиля пользователей	Ссылка
4.	Страница «мои подписки»	Ссылка
5.	Front-end: Базовая вёрстка страницы профиля; Базовая вёрстка страницы «мои подписки»	Страница профиля, страница «мои подписки»
6.	Диаграммы: Классов; Взаимодействия; Активности	-
7.	Аналитика: «Авторизация»; «Переход на страницу пользователя»; «Оценка желаний»	-

8.	Курсовая работа	-
9.	Презентация	-
10.	Настройка REST API: Account; UserPage; Followings;	ссылка
11.	Тестирование: Дымовое	-
12.	Классы – репозитории: User; Following;	User + Following
13.	Список распределения задач	-